نظام الملفات الممتد

Extended File System

Ext4 Disk Layout

تخطيط نظام الملفات على القرص

مسوّدة 2

جدد في محرم 1441/ سبتمبر أيلول 2019

يُوزَع مجانًا ولا يُباع

محتويات

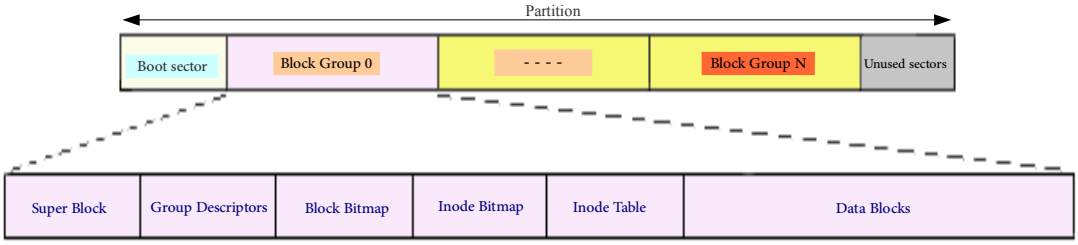
تخطيط Layout

رغم وجود أنظمة ملفات حرة كثيرة [135] معظم توزيعات جنو لينكس وأنظمة أخرى حاليا تستخدم أخر إصدار من عائلة EXT في إدارة الملفات على الأقراص، وإلى جانب جذور نظام الملفات الأخرى [133] بنية البيانات الوصفية التالية مستوحاة من تصميم نظام ملفات يونكس UFS/FFS. المساحة في EXT تبدأ بمنطقة محجوزة اختيارية عند الكتلة 0، ثم بقية نظام الملفات مجزأ إلى متتالية كتل منطقية في مجموعات، جميعها تملك نفس عدد الكتل باستثناء المجموعة الأخيرة. هذه المجموعات تشبه ما يسمى المجموعات الأسطوانية في نظام ملفات UFS/FFS، لكن الكتل في EXT لا ترتبط بالتخطيط الفيزيائي على القرص [18].

محصص الكتل سيحاول الحفاظ على كتل الملف ضمن نفس المجموعة، لخفض التجزئة والتقليل من مدة البحث أو السعي (حركة رأس القرص الصلب).

حجم مجموعة الكتل يحدد في الكتلة العليا [112] superblock في حقل s_blocks_per_group الذي يمكن حسابه كالتالي block_size_in_bytes * 8 ولأن حجم الكتلة المبدئية هو 4 كيلوبايت إذن كل مجموعة ستتضمن 32،768 كتلة (8 × 4096)، بطول 128 ميغابايت (32768 × 4096). عدد مجموعات الكتل سيكون حاصل قسمة حجم الجهاز device على حجم مجموعة الكتل block group، مثال:3908091 ÷ 32768 = 119.265472412 أي 120 مجموعة. كل واحدة بحجم 32 كيلوبايت باستثناء مجموعة الكتل الأخيرة.

في نظام ملفات EXT4 جميع الحقول تكتب إلى القرص بترتيب نهوي صغير- باستثناء حقول نظام قيد الحوادث (jbd2) التي تكتب بترتيب نهوي كبير- عموما نفس التصميم ينطبق على أنظمة الملفات ext2/3 . رغم أن حقولها أقصر ولا تدعم جميع ميزات ext4 [104] أيضا واستناداً إلى ملاحظات المؤلف djwong. تعريفات هياكل البيانات يجب أن تكون مجاربة لإصدارات لينكس Linux 4.18 وحزمة e2fsprogs-1.44 [103]



معلومات نظام الملفات الأساسية تخزن في الكتلة العليا super block، في بداية نظام الملفات (القسم). ومضمون الملفات يخزن في كتل البيانات وتقريبا التخطيط المعياري لمجموعة الكتل سيكون بشكل التالي :

حشو في / قبل بداية المجموعة 0 Group 0 Padding	الكتلة العليا ext4 Super Block	توصيفات المجموعات Group Descriptors	كتل محجوزة للتوصيفات المجموعات Reserved GDT Blocks [50]	مصفوفة ثنائية لكتل البيانات Data Block Bitmap	مصفوفة ثنائية لمؤشرات الفهرسة Inode Bitmap	جدول مؤشرات الفهرسة Inode Table	كتل بيانات Data Blocks
بايت 1024	كتلة 1	كتل معدودة	كتل معدودة	كتلة 1	كتلة 1	كتل معدودة	كتل كثيرة جدا

أول 1024 بايت في أو قبل مجموعة الكتل 0، محجوزة للسماح بتثبيت برامج وشفرات إقلاع x86، ولذلك كتلة superblock تبدأ عند إزاحة بايت 1024 (أي كان حجم الكتلة)، لكن إذا كان حجم الكتلة 1024 بايت، superblock ستكون في الكتلة 1.

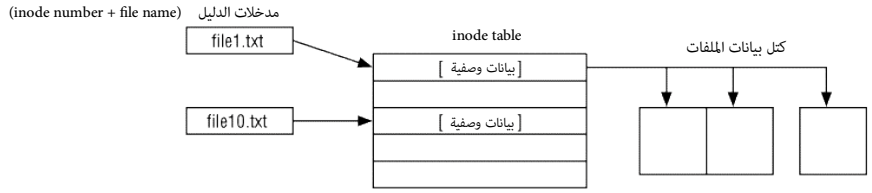
منطقة(باستثناء مجموعة الكتل 0) حشوة 1024 بايت لا توجد في/قبل مجموعات الكتل الأخرى.

عند تهيئة النظام أول مرة، أداة mkfs ستخصص مساحة من أجل كتل "reserve GDT block" بعد group descriptors وقبل بداية block bitmaps للسماح بتوسع نظام الملفات مستقبلا مبدئيا، يمكن مضاعفة حجم نظام الملفات بمقدار 1024 مرة[50] في Group Descriptors حقول grp.bg_inode_table_* سوف تحدد موقع inode table الذي هو نطاق متواصل من الكتل التي تكفي لاحتواء قيمة: sb.s_inodes_per_group * sb.s_inode_size.

بالنسبة للترتيب العناصر، الثابت هو أن كتلة super block وكتل group descriptor table إن وجدت، ستكون في بداية مجموعة الكتل. أما كتل bitmaps و inode table فيمكن أن تكون في أي مكان، واحتمال كبير أن تأتي bitmaps بعد inode table، أو كلاهما يكون في مجموعات كتل مختلفة (راجع ميزة flex_bg). المساحة المتبقية ستكون من أجل كتل بيانات الملفات file data blocks، وربط الكتل الغير مباشر (ة) indirect block maps، أو كتل شجرة المديات extent tree، والخصائص الممتدة للملفات extended attributes [121]..

البيانات الوصفية (المصاحبة) لكل ملف و دليل (مجلد) تخزن في سجل ملف يدعى Inode (مؤشر فهرسة!) [115][116][01] الذي يملك حجم ثابت ويقع في جدول مؤشرات الفهرسة inode table. وهناك جدول واحد في كل مجموعة كتل.

اسم الملف file name يخزن في مدخلة الدليل [117] التي تقع في الكتل المخصصة للدليل الأم الخاص بالملف. ومدخلات الدليل عبارة عن هياكل بسيطة تتضمن اسم الملف ومؤشر إلى مدخلة مؤشر فهرسة الملف (inode number).



العلاقة بين مدخلة الدليل Directory entry، ومؤشر الفهرسة inode، والكتل blocks

الكتل Blocks

نظام ملفات ext4 يوزع مساحة التخزين على وحدات من الكتل. والكتلة عبارة عن مجموعة قطاعات[41] بين 1 و 64 كيلوبايت (تشبه مفهوم clusters في FAT/NTFS) عدد القطاعات يجب أن يكون عدد صحيح أس اثنين بدورها الكتل ستجمع في وحدات أكبر تدعى مجموعات الكتل block groups. حجم الكتلة [62] يحدد في زمن mkfs، والقيمة المبدئية 4 كيلوبايت. علما أن المستخدم سيواجه مشاكل في وصل نظام الملفات إذا كان حجم الكتلة أكبر من حجم الصفحة الذاكرية (مثلا 64 كيلوبايت على نظام i386 الذي يملك فقط صفحات 4k) في العادة، نظام الملفات يمكن أن يتضمن 2^{32} كتلة؛ وفي حالة تمكين ميزة '64bit' نظام الملفات يمكن أن يملك 2^{64} كتلة.

حدود نظام الملفات القصوى

عنصر		1 كيلوبايت	2 كيلوبايت	4 كيلوبايت	64 كيلوبايت	نمط / نظام	
						32-بت	64-بت
كتل	Blocks	2^{32}	2^{32}	2^{32}	2^{32}	√	
		2^{64}	2^{64}	2^{64}	2^{64}		√
مؤشرات فهرسة (سجلات ملفات، عقد ملفات!)	Inodes	2^{32}	2^{32}	2^{32}	2^{32}	√	√
حجم نظام الملفات	File System Size	4 تيرابايت	8 تيرابايت	16 تيرابايت	256 بتابايت	√	
		16 زيتابايت	32 زيتابايت	64 زيتابايت	1 يوتابايت		√
عدد الكتل لكل مجموعة كتل	Blocks Per Block Group	8,192	16,384	32,768	524,288	√	√
عدد مؤشرات الفهرسة لكل مجموعة كتل	Inodes Per Block Group	8,192	16,384	32,768	524,288	√	√
حجم مجموعة الكتل	Block Group Size	8 ميغابايت	32 ميغابايت	128 ميغابايت	32 جيجابايت	√	√
عدد الكتل لكل ملف، مديات	Blocks Per File, Extents	2^{32}	2^{32}	2^{32}	2^{32}	√	√
عدد الكتل لكل ملف، ربط كتل (تعيين كتل)	Blocks Per File, Block Maps	16,843,020	134,480,396	1,074,791,436	* 4,398,314,962,956	√	√
حجم الملف، مديات	File Size, Extents	4 تيرابايت	8 تيرابايت	16 تيرابايت	256 تيرابايت	√	√
حجم الملف، ربط كتل (تعيين كتل)	File Size, Block Maps	16 جيجابايت	256 جيجابايت	4 تيرابايت	256 تيرابايت	√	√

* (القيمة فعليا ستكون 2^{32} بسبب حجم الحقل المحدود)

الملفات التي تستخدم ربط الكتل (أي لا تستخدم extents) يجب وضعها في حدود أول 2^{32} كتلة من نظام الملفات والملفات التي تستخدم المديات يجب وضعها في حدود أول 2^{48} كتلة من نظام الملفات. وليس واضح ما يمكن أن يحدث مع أنظمة الملفات الأكبر.

مجموعات الكتل المرنة ! Flexible Block Groups

ميزة جديدة في نظام ملفات ext4، تسمح بربط أو جمع عدة مجموعات كتل (مبدئيا ستكون 16) متجاورة/متماسة في مجموعة منطقية واحدة؛ بحيث مساحات كل من bitmaps [32] و inode table في أول مجموعة كتل من كل مجموعة مرنة flex_bg تتمدد لتشمل البيانات الوصفية inode tables و bitmaps من جميع مجموعات الكتل الأخرى في flex_bg. في المثال التالي كل مجموعة مرنة flex_bg بحجم 16 مجموعة، مع موجود 120 مجموعة كتل (أي 8 مجموعات مرنة) [55]:

Flexible Block Groups	120 Block Groups inside Flexible Block Groups															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119								

كل صف يعتبر مجموعة كتل مرنة Flexible Block Group وكل خلية تعتبر مجموعة كتل Block Group. مثلا مجموعة الكتل 0 تتضمن البيانات الوصفية التالية بالترتيب (للمجموعات 0-15) المساحة المتبقية ستشغلها بيانات الملفات.

وكما تلاحظ نسخ Superblock و group descriptors ستكون فقط في المجموعات 1، 3، 5، 7، 9، 25، 27، 49، 81.

```
Group 0: (Blocks 0-32767) [ITABLE_ZEROED]
  Checksum 0x9a88, unused inodes 8131
  Primary superblock at 0, Group descriptors at 1-1
  Reserved GDT blocks at 2-955
  Block bitmap at 956 (+956), Inode bitmap at 972 (+972)
  Inode table at 988-1496 (+988)
  23630 free blocks, 8133 free inodes, 2 directories, 8133 unused inodes
  Free blocks: 9138-32767
  Free inodes: 12-8144
```

مجموعة الكتل الأولى في كل مجموعة كتل مرنة تتضمن البيانات الوصفية: Block Bitmaps و inode Bitmaps و inode Tables من جميع مجموعات الكتل داخل المجموعة المرنة. بقية المجموعات ستكون كتل بيانات. لاحظ أن مجموعة الكتل المرنة 8 (أي الأخيرة) تتضمن فقط 8 مجموعات كتل: 119، 118، 117، 116، 115، 114، 113، 112. تأثير أو فائدة هذا، ستكون في محلية البيانات الوصفية التي تعني تسريع تحميل (البيانات الوصفية) بالإضافة إلى التخزين المتواصل على القرص للملفات الكبيرة.

وسيكون هناك دائما نسخ احتياطية من superblock و group descriptors في مجموعات الكتل، [86] (sparse_super) حتى في حالة تمكين ميزة flex_bg. وعدد مجموعات الكتل التي تشكل flex_bg سيكون ناتج: 2 ^ sb.s_log_groups_per_flex .

الجمع بين ميزة flex_bg و ميزة sparse_super، ينتج عنه مجموعات كتل معظمها كتل بيانات. هذا يتحقق إذا كان ترتيب مجموعات الكتل المرنة flex_bg أس العدد اثنين (عدد زوجي/ أي يقبل القسمة على عدد مجموعات الكتل في flex_bg) ومجموعات sparse_super، أس العدد 3، 5، و 7. ...(أي عدد فردي) بذلك، المجموعة 0 block group ستكون مجموعة الكتل الوحيدة التي تتضمن: Primary superblock و group descriptor table و GDT و bitmaps و inode tables. وبسبب ميزة sparse_super، المجموعة 1 block group ستتضمن superblock و group descriptor table. ومن ثم واعتمادا على حجم مجموعات flex_bg، المجموعة 16 block group يمكن أن تبدأ مجموعة كتل مرنة جديدة (أنظر أعلاه) تتضمن bitmaps و inode table. بقية المجموعات في المجموعة المرنة ستكون كتل بيانات فقط، باستثناء مجموعات الكتل التي تتضمن نسخ superblock و group descriptor table بسبب sparse_super.

مجموعات الكتل الوصفية !

Meta Block Groups

بدون الخيار META_BG [95] لاعتبارات أمنية، يتم حفظ جميع نسخ توصيفات مجموعات كتل group descriptors في أول مجموعة كتل. وبالنظر إلى حجم مجموعة الكتل المبدئي 128 ميغابايت (2^{27} بايت) وحجم توصيف المجموعات 64-بايت، يمكن للنظام ملفات ext4 في الغالب امتلك $2^{27} \div 64 = 2^{21}$ مجموعة كتل. هذا سيحد من حجم كامل نظام الملفات إلى $2^{21} \times 2^{27} = 2^{48}$ بايت أو 256 تيرابايت. حل هذا المشكلة !؟ كان باستخدام ميزة META_BG، الموجودة أيضا في ext3 (لينكس 2.6) والتي تقسم ext4 إلى مجموعات كتل وصفية كثيرة metablock groups. كل واحدة عبارة عن عنقود من مجموعات الكتل، التي هياكل توصيف مجموعاتها يمكن أن تخزن في كتلة واحدة على القرص. في نظام ملفات ext4 الذي يوظف حجم الكتلة 4 كيلوبايت، مبدئيا (قسم) مجموعة الكتل الوصفية الواحدة سيتضمن 64 مجموعة كتل أو 8 جيجابايت من مساحة القرص. ميزة META_BG سوف تحرك موقع group descriptors من مجموعة الكتل الأولى المكتظة لكامل نظام الملفات إلى أول مجموعة كتل من كل metablock group نفسها. والنسخ الاحتياطية ستكون في المجموعة الثانية والأخيرة من كل metablock group. هذا يرفع الحد الأقصى للمجموعات الكتل إلى 2^{32}، الذي يسمح بدعم نظام ملفات 512 بيتابايت.

في تغيير جديد، بدلا من صيغة نظام الملفات المخطط الحالي حيث كتلة superblock تتبعها تشكيلة متغيرة الطول من group descriptors، توضع superblock مع كتلة واحدة group descriptor في بداية مجموعة الكتل الأولى، والثانية، والأخيرة في مجموعة الكتل الوصفية و metablock group ستكون تجميع لمجموعات الكتل التي يمكن وصفها بواسطة كتلة واحدة group descriptor. وبما أن حجم بنية group descriptor هو 32 بايت، metablock group سوف تتضمن 32 مجموعة في أنظمة ملفات كتلة 1 كيلوبايت، أو تتضمن 128 مجموعة في أنظمة ملفات كتلة 4 كيلوبايت. ويمكن إنشاء أنظمة الملفات إما باستخدام التخطيط الجديد هذا group descriptor أو بعمل إعادة تحجم متصل لأنظمة الملفات الموجودة. وحقل s_first_meta_bg في superblock سوف يشير إلى أول مجموعة كتل تستخدم هذا التخطيط الجديد.

لا تنسى مراجعة الملاحظة الهامة حول BLOCK_UNINIT في فقرة المصفوفات الثنائية للمؤشرات الفهرسة وكتل البيانات block and inode bitmaps.

	128 block groups inside metablock groups																															
metablock groups	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
2	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
3	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

في هذا المثال، حجم الكتلة كان 1024 بايت، عدد الكتل 131072، كل مجموعة 1024 كتلة، كل مجموعة تتضمن 32 مجموعة بمجموع 128 مجموعة، لاحظ هنا استخدام المبرزان meta_bg و sparse_super ، لاحظ أيضا وجود Primary superblock في الكتلة 1 في المجموعة 0.

كل صف يعتبر مجموعة كتل وصفية Meta Block Group، وكل خلية تعتبر مجموعة كتل. لاحظ في هذه العينة التالية من مجموعة الكتل الوصفية الأولى (0) مجموعة الكتل 0 تتضمن بالترتيب البيانات الوصفية التالية:

Group 0: (Blocks 1-1024)
Primary **superblock** at 1, **Group descriptor** at 2
Block bitmap at 3 (+2), **Inode bitmap** at 4 (+3)
Inode table at 5-36 (+4)
975 free blocks, 245 free inodes, 2 directories
Free blocks: 50-1024
Free inodes: 12-256

Group 1: (Blocks 1025-2048)
Backup superblock at 1025, **Group descriptor** at 1026
Block bitmap at 1027 (+2), Inode bitmap at 1028 (+3)
Inode table at 1029-1060 (+4)
988 free blocks, 256 free inodes, 0 directories
Free blocks: 1061-2048
Free inodes: 257-512

Group 2: (Blocks 2049-3072)
Block bitmap at 2049 (+0),Inode bitmap at 2050 (+1)
Inode table at 2051-2082 (+2)
990 free blocks, 256 free inodes, 0 directories
Free blocks: 2083-3072
Free inodes: 513-768

Group 3: (Blocks 3073-4096)
Backup superblock at 3073
Block bitmap at 3074 (+1),Inode bitmap at 3075 (+2)
Inode table at 3076-3107 (+3)
989 free blocks, 256 free inodes, 0 directories
Free blocks: 3108-4096
Free inodes: 769-1024

Group 31: (Blocks 31745-32768)
Group descriptor at 31745
Block bitmap at 31746 (+1),Inode bitmap at 31747 (+2)
Inode table at 31748-31779 (+3)
989 free blocks, 256 free inodes, 0 directories
Free blocks: 31780-32768
Free inodes: 7937-8192

نسخ من Superblock ستكون فقط في المجموعات 1، 3، 5، 7، 9، 25، 27، 49، 81، 125. لاحظ أيضا أن مجموعة الكتل الأولى والثانية والأخيرة في كل مجموعة كتل وصفية تبدأ مع Group descriptor إلى جانب البيانات الوصفية الأخرى.

بقية المجموعات تتضمن كتل بيانات إلى جانب البيانات الوصفية Block bitmap, Inode bitmap, Inode table. هذه عينة أخرى من مجموعة الكتل الوصفية الثانية (1):

Group 32: (Blocks 32769-33792)
Group descriptor at 32769
Block bitmap at 32770 (+1), Inode bitmap at 32771 (+2)
Inode table at 32772-32803 (+3)
989 free blocks, 256 free inodes, 0 directories
Free blocks: 32804-33792
Free inodes: 8193-8448

Group 33: (Blocks 33793-34816)
Group descriptor at 33793
Block bitmap at 33794 (+1), Inode bitmap at 33795 (+2)
Inode table at 33796-33827 (+3)
989 free blocks, 256 free inodes, 0 directories
Free blocks: 33828-34816
Free inodes: 8449-8704

Group 34: (Blocks 34817-35840)
Block bitmap at 34817 (+0), Inode bitmap at 34818 (+1)
Inode table at 34819-34850 (+2)
990 free blocks, 256 free inodes, 0 directories
Free blocks: 34851-35840
Free inodes: 8705-8960

Group 49: (Blocks 50177-51200)
Backup superblock at 50177
Block bitmap at 50178 (+1), Inode bitmap at 50179 (+2)
Inode table at 50180-50211 (+3)
989 free blocks, 256 free inodes, 0 directories
Free blocks: 50212-51200
Free inodes: 12545-12800

Group 63: (Blocks 64513-65536)
Group descriptor at 64513
Block bitmap at 64514 (+1), Inode bitmap at 64515 (+2)
Inode table at 64516-64547 (+3)
989 free blocks, 256 free inodes, 0 directories
Free blocks: 64548-65536
Free inodes: 16129-16384

(البيانات من ملف e2fsprogs-1.44.0/tests/m_meta_bg)

التهيئة المؤجلة لمجموعة الكتل !

Lazy Block Group Initialization

الأعلام الثالثة في واصف مجموعة الكتل هي إحدى الميزات الجديدة في EXT4. التي تمكن أداة بناء نظام الملفات mkfs من تخطي تهيئة الأجزاء الأخرى من البيانات الوصفية لمجموعة الكتل. خصوصا، أعلام INODE_UNINIT و BLOCK_UNINIT التي تعني أن inode / block bitmaps لتلك المجموعة يمكن حسابها وبالتالي كتل bitmap على القرص لن يتم تهيئتها uninitialized. عموما هذا هو الحال مع مجموعة الكتل الشاغرة empty block group أو مجموعة الكتل التي تتضمن فقط بيانات وصفية ذات الموقع الثابت للمجموعة الكتل. علم INODE_ZEROED يعني أن inode table قد تم تهيئته؛ أداة mkfs ستلغي تعين هذا العلم وتعتمد على النواة في تهيئة inode tables في الخلفية.[47]

عدم كتابة الأصفار إلى inode table و bitmaps يعني تسريع عمل أداة mkfs عند إنشاء نظام الملفات.وتسريع e2fsck عند فحص النظام. لاحظ أن علم هذه الميزة في superblock هو RO_COMPAT_GDT_CSUM لكن في خرج dumpe2fs يظهر "uninit_bg" الذي له نفس المعنى.

مؤشرات فهرسة خاصة

Special inodes

نظام ملفات EXT4 يحتفظ ببعض مؤشرات الفهرسة inodes الخاصة والمخفية التي تظهر في الجدول التالي:

رقم مؤشر الفهرسة	ثابت / معامل	غرض
0		مؤشر الفهرسة 0 (هذا لا يوجد أصلا / لا يستخدم) [39]
1	EXT4_BAD_INO	مؤشر فهرسة قائمة الكتل المعيبة (كتل تالفة) Allocated over bad blocks by mke2fs / e2fsck -c
2	EXT4_ROOT_INO	مؤشر فهرسة الدليل الجذر
3	EXT4_USR_QUOTA_INO	مؤشر فهرسة ملف حصة المستخدم (سابقا كان لأجل EXT2_ACL_IDX_INO (ACL index inode
4	EXT4_GRP_QUOTA_INO	مؤشر فهرسة ملف حصة المجموعة (سابقا كان لأجل EXT2_ACL_DATA_INO (ACL data (الحصص النسبية للقرص) [13]
5	EXT4_BOOT_LOADER_INO	مؤشر فهرسة محمل الإقلاع (شفرة الإقلاع) من أجل إخفاء أو حماية stage2 loaders في نظام الملفات (غير مستخدم)
6	EXT4_UNDEL_DIR_INO	مؤشر فهرسة دليل استرجاع الملفات المحذوفة / يبدو من أجل وظيفة undeletion التي لم تطبق أبدا
7	EXT4_RESIZE_INO	مؤشر فهرسة توصيفات المجموعات المحجوزة / إعادة تحجيم نظام الملفات ("resize inode") [50] [96]
8	EXT4_JOURNAL_INO	مؤشر فهرسة قيد الحوادث [113] (Journal inode)
9	EXT4_EXCLUDE_INO	مؤشر فهرسة لحفظ كتل "exclude bitmap"(يستخدمه نظام ملفات Next3 الذي هو نسخة معدلة من ext3) (مؤشر الفهرسة 9 كان سابقا من أجل EXT4_ORPHAN_INO)
10	EXT4_REPLICA_INO	مؤشر فهرسة النسخ المتماثلة، يستخدم مع بعض ميزات الغير رسمية ؟ (ext4 metadata replication out of tree in a Google patch)
11	EXT4_GOOD_OLD_FIRST_INO	أول مؤشر فهرسة تقليدي غير محجوز. عادة يكون الدليل lost+found (راجع حقل s_first_ino في superblock) [105]

سياسة توزيع الكتل ومؤشرات الفهرسة

Block and Inode Allocation Policy

لقد أدرك المطورون في ext4، أن محلية البيانات ميزة مطلوبة جداً في نظام ملفات. فالاحتفاظ بالكتل ذات الصلة قرب بعضها البعض على القرص الدوار (القرص الثابت)، يخفف من حركة القرص والمشغل للوصول إلى كتلة البيانات. وبالتالي تسريع وحدات إدخال وإخراج القرص disk IO. طبعا على أقراص SSD لا توجد أجزاء متحركة، ولكن المحلية يمكن أن تزيد في حجم كل طلب ناقل بيانات مع تخفيض العدد الإجمالي للطلبات. هذه المحلية يمكن أن يكون لها أيضا تأثير الكتابات المكثفة على كتلة مسح واحدة erase block، التي يمكن أن تسرع إعادة كتابة الملفات بشكل ملحوظ. ومن ثم خفض التجزئة سيكون مفيد حيث أمكان.

الآلية أو الأداة الأولى التي يستخدمها نظام ملفات ext4 في معركته مع التجزئة تدعى محصص الكتل المتعددة [126] mballoc. هذا الأخير، عند إنشاء الملف أول مرة، يخصص 8 كيلوبايت من مساحة القرص للملف على افتراض أن المساحة سيتم كتابتها قريبًا. وعند غلق الملف، يتم تحرير الحصص الغير مستخدمة. لكن إذا كان التخمين صحيح (كما في الكتابات الكاملة للملفات الصغرى) حينذاك تدون بيانات الملف في مدى متعدد الكتل.

الآلية الثانية: التخصيص المتأخر للكتل delayed allocation [28]، في هذا المخطط، عندما يحتاج الملف إلى المزيد من الكتل لاستيعاب كتابات الملف، نظام الملفات يؤجل تقرير الموضع الصحيح على القرص حتى يتم كتابة كافة بيانات الصوان الملوثة (معدلة!) dirty buffer[26][125] إلى القرص. والتنفيذ إلى الموضع المعين سيكون للضرورة فقط (حتى تحين مهلة التنفيذ أو يتم استدعاء ()sync، أو تستهلك النواة الذاكرة). على أمل أن يتخذ نظام الملفات قرارات أفضل بشأن الموقع.

الآلية الثالثة: نظام الملفات يحاول الإبقاء على كتل الملف ضمن نفس مجموعة الكتل حيث يوجد inode [115][116] هذا سيخفض من مدة السعي عندما يتحتم على نظام الملفات قراءة أولاً inode لمعرفة مكان تواجد كتل بيانات الملف ثم السعي إلى كتل بيانات الملف من أجل بدء عمليات I/O .

الآلية الرابعة: توضع جميع inodes في الدليل في نفس مجموعة الكتل حيث يوجد الدليل إن أمكان، على افتراض أن جميع الملفات في الدليل ذات صلة. وبالتالي اجتماعها سيكون مفيد.

الآلية الخامسة: يتم تجزئة وحدة التخزين على القرص إلى مجموعات من الكتل، كل واحدة 128 ميغابايت؛ هذه الحاويات الصغرى، كما سبق وأن ذكرنا للحفاظ على محلية البيانات. لكن، هناك ميزة غريبة ولكن متعمدة -- عند إنشاء دليل في الدليل الجذر، محصص inodes يتفحص مجموعات الكتل ويضع ذلك الدليل في مجموعة الكتل المحملة الأخف التي يمكن أن يجدها. هذا يساعد في نشر الأدلة عبر القرص؛ بما أن الملفات / الأدلة من نوع blobs [36] على المستوى الأعلى تشغل مجموعة كتل واحدة، المحصّص سينتقل ببساطة إلى مجموعة الكتل التالية.ظاهريًا هذا التخطيط يوازن التحميل على مجموعات الكتل. رغم أن، الكاتب يشك في نفس المعاملة مع الأدلة الواقعة قرب نهاية القرص الدوار. (راجع أيضا: خوارزمية محصص الكتل أورلوف) طبعا، إذا فشلت جميع هذه الآليات. يمكنك دائما استخدام أداة e4defrag في الغاء تجزئة الملفات.

تدقيق المجموع

Checksum

تدقيق المجموع موجود في أهم هياكل بيانات ext4 و jbd2 [48] منذ 2012. علم الميزة هو metadata_csum. وخوارزمية تدقيق المجموع المطلوبة ستشير إليها كتلة superblock، لكن (اعتبارا من أكتوبر 2012) الخوارزمية الوحيدة المدعومة هي crc32c.. بعض هياكل البيانات لا تملك مساحة كافية لاحتواء كامل تدقيق مجموع 32-بت، لذلك تخزن فقط 16 بت المنخفضة. تمكين ميزة 64bit سيرفع حجم بنية البيانات وبالتالي يمكن تخزين كامل تدقيق المجاميع 32-بت مع هياكل بيانات كثيرة. على أية حال، أنظمة 32-بت الموجودة لا يمكنها استعمال نمط 64bit، على الأقل ليس بدون استخدام الرقع resize2fs.في أنظمة الملفات الحالية يمكن إضافة تدقيق المجموع بتنفيذ tune2fs -O metadata_csum على الجهاز الأساسي. إذا صادف tune2fs كتل دليل لا تملك مساحة كافية لإضافة تدقيق المجموع، سيطلب من المستخدم تنفيذ e2fsck -D لإعادة بناء الأدلة مع تدقيق المجاميع. هذا أيضا له فائدة في إزالة التجزئة الداخلية من ملفات الدليل وإعادة حفظ توازن فهارس htree[25]إذا تجاهل المستخدم هذه الخطوة، الأدلة لن تكون محمية بتدقيق المجموع.

عناصر البيانات (أو المكونات) التي تدخل في كل نوع من تدقيق مجموع. دالة تدقيق المجموع تحددها superblock (وستكون crc32c منذ أكتوبر 2013) باستثناء ما في الملاحظة.

وصف	بيانات وصفية		حجم / نوع	
كامل كتلة superblock حتى حقل تدقيق المجموع. معرف UUID سيكون في داخل كتلة superblock	كتلة عليا	Super block	4	__le32
UUID + كامل كتلة MMP حتى حقل تدقيق المجموع	حماية من تعدد الوصل	MMP	4	__le32
UUID + كامل كتلة الخصائص الممتدة (حقل تدقيق المجموع سيكون صفر)	خصائص ممتدة	Extended ttributes	4	__le32
UUID + رقم inode + رقم توليد inode + كتلة الدليل حتى المدخلة المزيفة التي تنطوي على حقل تدقيق المجموع	مدخلات أدلة	Directory Entries	4	__le32
UUID + رقم inode + رقم توليد inode + جميع المدبات الصالحة + ذيل HTREE. (حقل تدقيق المجموع سيكون صفر)	عقد شجرة HTREE	HTREE Nodes	4	__le32
UUID + رقم inode + رقم توليد inode + كامل كتلة المدى حتى حقل تدقيق المجموع	مدبات	Extents	4	__le32
UUID + كامل المصفوفة الثانية [32]. تدقيق المجاميع يخزن في group descriptor، ومقتطع في حالة كان حجم توصيف المجموعة 32 بايت (أي، 64bit^)	مصفوفات ثنائية	Bitmaps	4 / 2	__le16_أو _le32
UUID + رقم inode + رقم توليد inode + كامل inode. (حقل تدقيق المجموع سيكون صفر) كل inode يملك تدقيق مجموع خاص	مؤشرات الفهرسة	Inodes	4	__le32
في حالة metadata_csum_(UUID+رقم المجموعة+كامل التوصيف) وفي حالة gdt_csum_(UUID+رقم المجموعة+كامل توصيف)crc16. في جميع الحالات تخزن فقط 16 بت المنخفضة	توصيف المجموعات	Group Descriptors	2	__le16

تخصيص الكتل الكبيرة

Bigalloc

حتى الآن، حجم الكتلة المعمول به في ext4 هو 4 كيلوبايت (4096 بايت)، هذا يوافق حجم الصفحة الشائع والمدعوم على معظم العتاد مع وحدة إدارة الذاكرة MMU. وهذا من حسن الحظ، لأن شفرة ext4 لا يمكنها التعامل مع حجم الكتلة الذي يتعد حجم الصفحة الذاكرية. على أية حال، تخصيص كتل للقرص بحجم وحدات من عدة كتل. مطلوب في النظم التي تستخدم الملفات الكبيرة جدا، لخفض التجزئة وفوقانية البيانات الوصفية [15]. هذه القدرة توفرها ميزة bigalloc.. مدير النظام يستطيع تعيين حجم عنقود الكتل زمن mkfs (المخزن في حقل s_log_cluster_size في superblock)؛ بعد ذلك، المصفوفات الثنائية للكتل ستتعقب العناقيد. وليس الكتل المنفردة. هذا يعني أن مجموعات الكتل يمكن أن تكون بحجم عدة حبجابايت (بدلا من 128 ميغابايت فقط)؛ على أية حال، وحدة التخصيص الأدنى ستصبح العنقود، وليس الكتلة، حتى مع الأدلة.

في هذا الشأن، كان لدى تاوباو TaoBao مجموعات رقع لتوسع "في استخدام وحدات من العناقيد بدلاً من الكتل" في شجرة المدبات [120] مع أنه ليس واضح أين ذهبت تلك الرقع-التي تحولت إلى "extent tree v2" تلك الشفرة لم تظهر منذ مايو 2015.

البيانات المضمنة / البيانات المباشرة — Inline Data

هذه الميزة [140] تمكن بيانات الملف الصغيرة جدا من التناسب بسهولة داخل inode، (نظريا) هذا يخفض من استهلاك كتل القرص (في أنظمة الملفات الاعتيادية هذه يمكن أن يوفر ما يصل إلى 1-3% من مساحة) ويخفض من زمن السعي. إذا كان الملف أصغر من 60 بايت، تخزن البيانات داخليا في inode.i_block.وإن كانت بقية الملف تتناسب داخل مساحة الخاصية الممتدة، يمكن إيجادها كخاصية ممتدة "system.data" ضمن متن مؤشر الفهرسة ("ibody EA") هذا طبعا سيحد من كمية الخصائص الممتدة التي يمكن إلحاقها بمؤشر الفهرسة. إذا حجم البيانات تجاوز i_block + ibody EA، تخصص كتلة اعتيادية وينقل المحتوى إليها.

إن لم تستخدم xattrs يمكن تخزين ما يصل إلى 160 بايت من البيانات في مؤشر فهرسة 256 بايت (منذ يونيو 2015، حين كان حقل i_extra_isize بحجم 28 بايت) قبل ذلك، كان الحد هو 156 بايت بسبب الاستخدام الغير فعال لمساحة inode .

ميزة البيانات المضمنة inline data تتطلب تواجد خاصية ممتدة extended attribute من أجل "system.data" حتى وإن كانت قيمة الخاصية attribute value بطول الصفر.

```
Inode: 12   Type: directory    Mode:  0755   Flags: 0x10000000
Generation: 3089239889    Version: 0x00000000:00000002
User:     0   Group:     0   Size: 60
File ACL: 0    Directory ACL: 0
Links: 3   Blockcount: 0
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x5bcb759e:a9872534 -- Sat Oct 20 20:36:14 2018
 atime: 0x5bcb759e:a9872534 -- Sat Oct 20 20:36:14 2018
 mtime: 0x5bcb759e:a9872534 -- Sat Oct 20 20:36:14 2018
crtime: 0x5bcb759e:a9872534 -- Sat Oct 20 20:36:14 2018
Size of extra inode fields: 32
Extended attributes:
  system.data (0)
Inode checksum: 0x68fa51fe
Size of inline data: 60
```

(EXT4_INLINE_DATA_FL)

هذا للدليل يتضمن دليل آخر، الدليل يخزن 60 بايت من inline data ولا يملك ميزة extent ويملك ميزة INLINE_DATA في inode

الأدلة المضمنة / الأدلة المباشرة — Inline Directories

أول 4 بايت من i_block ستكون رقم inode للدليل الأم. يتبع ذلك، مساحة 56-بايت من أجل مصفوفة من مدخلات الدليل [117]؛ (راجع struct ext4_dir_entry). إذا كانت خاصية "system.data" موجودة في متن inode، قيمة EA تكون كذلك مصفوفة من struct ext4_dir_entry. لاحظ لأجل الأدلة المضمنة، مساحة i_block و EA تعامل ككتل منفصلة dirent ؛ مدخلات الدليل لا يمكنها الاتساع للإثنيان.

مدخلات الدليل المضمنة لا يتم التحقق من تدقيق مجموعها، لأن تدقيق مجموع inode سيحمي جميع محتويات البيانات المضمنة inline data.

قيم الخصائص الممتدة الكبيرة — Large Extended Attribute Values

لتمكين نظام ملفات ext4 من تخزين قيم الخصائص الممتدة [121] التي لا تتناسب في inode أو في كتلة الخصائص الممتدة الملحقة ببنية inode، تستخدم الميزة EA_INODE التي تسمح بتخزين القيمة في كتل بيانات مؤشر فهرسة الملف الاعتيادي. هذه "EA inode" ترتبط فقط[40] من جهة فهرس أسماء الخصائص الممتدة extended attribute name index ولا يجب أن تظهر في مدخلة الدليل [117].

في inode، سوف يستخدم حقل i_atime في تخزين تدقيق مجموع قيمة xattr ؛ وحقل i_ctime / i_version في تخزين [38] تَعْداد المراجع 64-بت، هذا يسمح بمشاركة قيم xattr الكبيرة بين عدة مؤشرات فهرسة مالكة **multiple** owning inodes (عندما تتشارك الملفات في نفس الخصائص)، وللتوافق خلفيا [99] مع الإصدارات الأقدم من هذه الميزة، يمكن في حقل i_mtime / i_generation تخزين مرجع خلفي إلى رقم inode و i_generation للمؤشر فهرسة مالك واحد **one** owning inode (في الحالات التي لا يتم الرجوع فيها إلى EA inode من عدة inodes) للتأكد من أن EA inode هو الصحيح الذي يتم النفاذ إليه.

تقليديا توزيعات لينكس لا تملك شفرة اقلاع في نظام الملفات (وحدة التخزين) إلى جانب النواة. وتستخدم عوض ذلك شفرة الاقلاع الابتدائية في قطاع MBR. (راجع كتيب MBR و GPT)، رغم ذلك أول قطاعين (أي 1.024 بايت) من بايت 0 إلى بايت 1023 قبل بداية كتلة نظام الملفات العليا super block، ستكون محجوزة من أجل شفرة قطاع الإقلاع x86، لكنها لا تستخدم، وقد تتضمن بيانات مخفية! من بعض التطبيقات.

حجم القطاع	أقصى سعة	نوع التنصيب	إقلاع	
512×2^{32} بايت	2.2 تيرابايت	BIOS-MBR	شفرة قطاع الإقلاع 0 + قطاعات بعد MBR (عادة تكون على الأقل 31 كيلوبايت أي 62 قطاع)	
		UEFI-MBR		
512×2^{64} بايت [23]	9.4 زيتابايت	BIOS-GPT	BBP + GRUB 2	شفرة الإقلاع في PMBR / Hybrid MBR
		UEFI-GPT	مدير الإقلاع + ESP	

مثال على تخطيط قرص BIOS-GPT

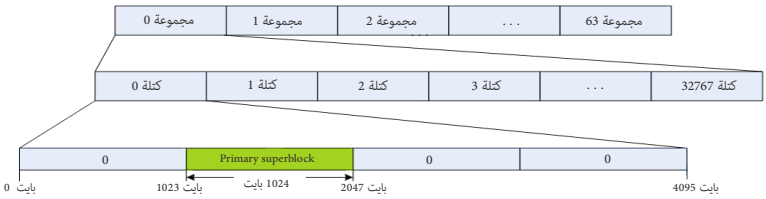
حجم مقترح	علم إقلاع	نوع قسم GUID	قسم	نقطة وصل
1 ميغابايت	نعم	BBP	/dev/sdx1	لا
وفقا لحجم القرص و RAM	لا	قسم الذاكرة الظاهرية/إبدال	/dev/sdx2	[SWAP]
بقية المساحة من تخصيص المستخدم	لا	لينكس	/dev/sdx3	/
	لا	(ملفات المستخدم)	/dev/sdx4	/home

مثال على تخطيط قرص UEFI-GPT

حجم مقترح	علم إقلاع	نوع قسم GUID	قسم	نقطة وصل
512 ميغابايت	نعم	ESP	/dev/sdx1	/boot
وفقا لحجم القرص و RAM	لا	قسم الذاكرة الظاهرية/إبدال	/dev/sdx2	[SWAP]
بقية المساحة من تخصيص المستخدم	لا	لينكس	/dev/sdx3	/
	لا	(ملفات المستخدم)	/dev/sdx4	/home

الكتلة العليا Super block

هذه الكتلة [112] من أجل التحكم في وحدة التخزين VCB (هذه الكتلة موجودة أيضا في أنظمة مثل مينكس و UFS وتشبه جدول FAT أو MFT في NTFS)، وتتضمن معلومات ضرورية لإقلاع نظام لينكس. لذلك توجد منها عدة نسخ احتياطية.لكن النسخة الأول Primary superblock فقط في أول كتلة يتم قراءتها عند وصل نظام الملفات (وحدة التخزين)، وتستخدم في الإقلاع. معلومات هذه الكتلة تسمح للمدير استخدام وصيانة النظام. نسخ من superblock و group descriptors ستكون فقط في المجموعة 0 و 1 وأس العدد 3، 5، 7، 9، 25، 27 إلى آخره. لكن في حالة تعطيل ميزة sparse_super [52][86] النسخ المكررة ستكون في جميع مجموعات الكتل. بينما تمكين الميزة sparse_super2 يسمح بوجود نسختين فقط من superblock و group descriptors. عادة تكون إحداها في بداية المجموعة block group #1، والأخرى في المجموعة الأخيرة في نظام الملفات. هذه الميزة الأخيرة تسمح بزيادة نسبة كتل البيانات المتماسة على القرص! للملفات. (راجع ميزة flex_bg).



مثال : موقع الكتلة العليا Primary superblock في نظام ملفات يملك 63 مجموعة كل مجموعة بحجم 32768 كتلة باستثناء المجموعة الأخيرة (هنا حجم الكتلة كان 4096 بايت)

كتلة superblock تقع دائما عند بايت 1024 من بداية وحدة التخزين ودائما بحجم ثابت 1024 بايت (مهما كان حجم الكتلة) في حالة قطاع 512 بايت، تبدأ عند الكتلة 2 LBA [51] وتشغل القطاعات 2 و 3 :

بنية الكتلة العليا Super block في struct ext4_super_block (المثال من نظام 32 بت)

```
dd if=/dev/sda1 bs=1024 count=1 skip=1 | hexdump -Cv
00000000  00 00 38 00 00 96 df 00  e6 2d 0b 00 22 69 48 00  |..8......-.."hH.|
00000010  41 3a 33 00 00 00 00 00  02 00 00 00 02 00 00 00  |A:3.............|
00000020  00 80 00 00 00 80 00 00  00 20 00 00 2a 73 b9 5a  |......... ..*s.Z|
00000030  09 7b be 5a 02 00 04 00  53 ef 01 00 01 00 00 00  |.{.Z....S.......|
00000040  79 5e ba 5a 00 00 00 00  00 00 00 00 01 00 00 00  |y^.Z............|
00000050  00 00 00 00 0b 00 00 00  00 01 00 00 3c 00 00 00  |............<...|
00000060  42 02 00 00 7b 00 00 00  91 2b 02 40 5b 79 47 c2  |B...{....+.@[yG.|
00000070  a4 8b 75 a6 e6 d5 cf f3  00 00 00 00 00 00 00 00  |..u.............|
00000080  00 00 00 00 00 00 00 00  2f 00 61 72 67 65 74 00  |......../.arget.|
00000490  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 fc 03  |................|
000004d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004e0  08 00 00 00 00 00 00 00  00 00 00 00 96 63 e8 05  |.............c..|
000004f0  1f d6 4f 28 8c 0d 03 33  f7 62 63 97 01 01 00 00  |..O(...3.bc.....|
00000100  0c 00 00 00 00 00 00 00  d0 9d 37 58 0a f3 02 00  |..........7X....|
00000110  04 00 00 00 00 00 00 00  00 00 00 00 ff 7f 00 00  |................|
00000120  00 80 68 00 ff 7f 00 00  01 00 00 00 ff ff 68 00  |..h...........h.|
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 08  |................|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 1c 00 1c 00  |................|
00000160  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000170  00 00 00 00 04 00 00 00  86 4e d9 0a 00 00 00 00  |.........N......|
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000230  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000250  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000260  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000270  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
[REMOVED]
000003f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

	حجم / نوع		إزاحة	رمز تذكري
عدد inodes الإجمالي في نظام الملفات [01] [56]	4	__le32	0x00 (00)	s_inodes_count
عدد الكتل الإجمالي في نظام الملفات [57]	4	__le32	0x04 (04)	s_blocks_count_lo
عدد الكتل المحجوزة لمنع شغل نظام الملفات [58] (32 بت المنخفضة) (سيكون من تخصيص المستخدم الجذر؛ ذو الصلاحيات العليا) (أنظر للحد 0x50)	4	__le32	0x08 (08)	s_r_blocks_count_lo
عدد الكتل الحرة [59] (الغير مخصصة)	4	__le32	0x0C (12)	s_free_blocks_count_lo
عدد inodes الحرة (الغير مخصصة) [60]	4	__le32	0x10 (16)	s_free_inodes_count
أول كتلة بيانات (تتضمن Super Block) (ستكون 0 بالنسبة لجميع أحجام الكتلة، باستثناء كتلة 1 كيلوبايت التي يجب أن تكون 1 على الأقل) [61]	4	__le32	0x14 (20)	s_first_data_block
حجم الكتلة [62] — 2 ^ (10 + s_log_block_size)	4	__le32	0x18 (24)	s_log_block_size
حجم العنقود [02] بعدد الكتل (2 ^ s_log_cluster_size). هذا في حالة تمكين ميزة bigalloc. ما عدا ذلك. s_log_cluster_size يجب أن يساوي s_log_block_size	4	__le32	0x1C (28)	s_log_cluster_size
عدد الكتل لكل مجموعة [63]	4	__le32	0x20 (32)	s_blocks_per_group
عدد العناقيد لكل مجموعة، في حالة تمكين ميزة bigalloc. ما عدا ذلك. s_clusters_per_group يجب أن يساوي s_blocks_per_group	4	__le32	0x24 (36)	s_clusters_per_group
عدد inodes لكل مجموعة [64]	4	__le32	0x28 (40)	s_inodes_per_group
زمن وصل نظام الملفات (وحدة التخزين) آخر مرة بعدد الثواني (توقيت يونكس)	4	__le32	0x2C (44)	s_mtime
زمن الكتابة إلى نظام الملفات آخر مرة بعدد الثواني (توقيت يونكس)	4	__le32	0x30 (48)	s_wtime
عدد مرات وصل نظام الملفات (وحدة التخزين) منذ آخر فحص fsck (منذ استخدام أداة فحص تماسك أو ثبات نظام الملفات!)	2	__le16	0x34 (52)	s_mnt_count
عدد مرات وصل نظام الملفات المطلوبة قبل فحص تماسك نظام الملفات fsck	2	__le16	0x36 (54)	s_max_mnt_count
رقم سحري! (التوقيع) [03] (لتأكيد !؟ وجود نظام الملفات EXT2/3/4 على وحدة التخزين) — 0xEF53 — EXT2_SUPER_MAGIC	2	__le16	0x38 (56)	s_magic

أختام زمنية

(أعلام) حالة نظام الملفات [65] القيم الصالحة ستكون:

نظام الملفات نظيف (إلغاء نقطة الارتباط صحيح / مفصول على نحو نظيف)	Unmounted cleanly	EXT4_VALID_FS	0x0001
أخطاء في نظام الملفات	Errors detected	EXT4_ERROR_FS	0x0002
استعادة inodes يتيمة (مؤشرات فهرسة معزولة)	Orphans being recovered	EXT4_ORPHAN_FS	0x0004

	2	__le16	0x3C (60)	s_errors

طريقة معالجة الخطأ [66] ستكون إحدى القيم الثلاثة التالية:

الاستمرار (تجاهل الخطأ)	Continue execution	EXT4_ERRORS_CONTINUE	1
إعادة وصل نظام الملفات في وضعية القراءة فقط	Remount fs read-only	EXT4_ERRORS_RO	2
خطأ فادح داخلي (نواة النظام في وضعية Panic)	Panic	EXT4_ERRORS_PANIC	3

مستوى مراجعة ثانوي (قيمة 16 بت) داخل مستوى المراجعة (راجع أيضا: نظام التحكم بالمراجعات)	2	__le16	0x3E (62)	s_minor_rev_level
زمن الفحص الأخير، بعدد الثواني (توقيت يونكس)	4	__le32	0x40 (64)	s_lastcheck
الفترة الأقصى بين الفحوص، بعدد الثواني	4	__le32	0x44 (68)	s_checkinterval

أختام زمنية

نظام التشغيل / هوية نظام التشغيل الذي عن طريقه تم إنشاء نظام الملفات على وحدة التخزين .

لينكس	Linux	EXT4_OS_LINUX	0
جنو هيرد (نواة)	Hurd	EXT4_OS_HURD	1
اسم نظام تشغيل من تطوير Rémy Card	Masix	EXT4_OS_MASIX	2
فري بي اس دي FreeBSD	FreeBSD	EXT4_OS_FREEBSD	3
النظم المبنية على BSD4.4-Lite	Lites	EXT4_OS_LITES	4

	4	__le32	0x4C (76)	s_creator_os

مستوى المراجعة. ستكون إحدى هذه:

صيغة أصلية	The good old (original) format	EXT4_GOOD_OLD_REV	0
صيغة 2 مع أحجام inode ديناميكية [19] وخصائص ممتدة....إلى آخره	V2 format w/ dynamic inode sizes	EXT4_DYNAMIC_REV	1

معرّف المستخدم، الذي يستطيع استخدام الكتل المحجوزة	نوع المعرّف : UID يشير إلى: المستخدم الجذري	EXT4_DEF_RESUID	0	s_def_resuid
معرّف المجموعة، التي تستطيع استخدام الكتل المحجوزة	نوع المعرّف : GID يشير إلى: المجموعة الجذرية	EXT4_DEF_RESGID	0	s_def_resgid

	2	__le16	0x50 (80)	s_def_resuid
	2	__le16	0x52 (82)	s_def_resgid

الحقول التالية فقط من أجل superblocks EXT4_DYNAMIC_REV. الاختلاف بين مجموعة الميزات المتوافقة [27] [104]compatible feature set والغير متوافقة incompatible feature set سيكون كالتالي: في حالة تعيين بت مجهول للنواة في الميزات الغير متوافقة، النواة يجب أن ترفض وصل نظام الملفات.. بينما في E2fsck يتم الغاء الميزة إذا كانت مجهولة للأداة سواء في الميزات المتوافقة أو الغير متوافقة

أول inode غير محجوز في نظام الملفات[67]	4	__le32	0x54 (84)	s_first_ino
حجم بنية inode- بعدد بايتات [04] [68] (راجع فقرة: Inode Size)	2	__le16	0x58 (88)	s_inode_size
رقم مجموعة الكتل التي تنتمي إليها هذه الكتلة العليا superblock [69]	2	__le16	0x5A (90)	s_block_group_nr
مجموعة أعلام الميزات المتوافقة [70] [85] — Compatible feature set flags :	4	__le32	0x5C (92)	s_feature_compat

كتل الدليل المخصصة مسبقا من أجل خفض التجزئة (عند إنشاء دليل جديد) (أنظر للحد 0xCD)	COMPAT_DIR_PREALLOC	0x0001
"imagic inodes" ليس واضح ماذا يفعل هذا العلم (لكنه يشير إلى وجود inodes خادوم AFS)	COMPAT_IMAGIC_INODES	0x0002
نظام ملفات مزود بقيد حوادث (إجراءات أو مجموعة updates في سجل دوري أو صوان حلقي ring buffer) [113]	COMPAT_HAS_JOURNAL	0x0004
دعم الخصائص الممتدة (inodes تملك خصائص ممتدة)	COMPAT_EXT_ATTR	0x0008
يملك كتل GDT من أجل توسع نظام الملفات (نظام الملفات يستطيع إعادة تحجيم نفسه إلى أقسام أكبر) [50] [95]	COMPAT_RESIZE_INODE	0x0010
يملك فهارس للدليل [40] (تستخدم شجرة hashed b-trees لتسريع عمليات البحث في الأدلة الكبيرة)	COMPAT_DIR_INDEX	0x0020

				0x0040	COMPAT_LAZY_BG	"Lazy BG" هذه الميزة تبدوا من أجل مجموعات الكتل الغير مهيئة؟ [05] uninitialized block groups
				0x0080	COMPAT_EXCLUDE_INODE	"Exclude inode" مؤشر فهرسة مرتبط بكتل exclude bitmap (غير مستخدم) (مستخدم في Next3)
				0x0100	COMPAT_EXCLUDE_BITMAP	للإشارة إلى وجود exclude bitmaps التي تتعقب الكتل المعينة إلى snapshot files [29] (غير مستخدم في النواة / e2fsprogs)
				0x0200	COMPAT_SPARSE_SUPER2	إذا تم تعيين هذا العلم، حقل s_backup_bgs سيشير إلى مجموعتان فقط من الكتل تتضمنان نسخ superblock [86]
				0x0400	COMPAT_ORPHAN_FILE	دعم إنشاء وحذف الملفات المعزولة ! (غير مستخدم في النواة / e2fsprogs)
s_feature_incompat	0x60 (96)	__le32	4	Incompatible feature set :		مجموعة أعلام الميزات الغير متوافقة [71] [85]
				0x0001	INCOMPAT_COMPRESSION	ضغط البيانات
				0x0002	INCOMPAT_FILETYPE	مدخلات الدليل تتضمن حقل نوع الملف (راجع ext4_dir_entry_2)
				0x0004	INCOMPAT_RECOVER	نظام الملفات يحتاج إلى استعادة Filesystem needs recovery
				0x0008	INCOMPAT_JOURNAL_DEV	نظام الملفات يملك جهاز قيد حوادث منفصل
				0x0010	INCOMPAT_META_BG	مجموعات الكتل الوصفية (راجع الميزة: Meta Block Groups) [95]
				0x0040	INCOMPAT_EXTENTS	الملفات في نظام الملفات تستخدم المديات (دعم extents) [120]
				0x0080	INCOMPAT_64BIT	تمكين حجم نظام الملفات 2^{64} كتلة (16 تيرابايت)
				0x0100	INCOMPAT_MMP	حماية نظام الملفات من الوصل المتعدد MMP. (غير مطبق) (راجع فقرة MMP)
				0x0200	INCOMPAT_FLEX_BG	مجموعات الكتل المرنة. (راجع الميزة: Flexible Block Groups)
				0x0400	INCOMPAT_EA_INODE	inodes يمكن استخدامها في تخزين قيم الخصائص الممتدة الكبيرة (راجع الميزة EA INODE)
				0x1000	INCOMPAT_DIRDATA	بيانات في مدخلة الدليل dirent (غير مطبق؟)
				0x2000	INCOMPAT_CSUM_SEED	بذرة [24] تدقيق مجموع البيانات الوصفية مخزنة في الكتلة العليا superblock [06]
				0x4000	INCOMPAT_LARGEDIR	دليل كبير > 2 جيجابايت أو مستوى 3 في شجرة Htree [07]
				0x8000	INCOMPAT_INLINE_DATA	تضمين بيانات في inode
				0x10000	INCOMPAT_ENCRYPT	وجود inodes مشفرة على نظام الملفات
				0x20000	INCOMPAT_CASEFOLD	دعم ترميز المحارف على مستوى نظام الملفات من أجل الأدلة في حالة تعيين casefold (مستخدم في e2fsprogs-1.45.2)
s_feature_ro_compat	0x64 (100)	__le32	4	Readonly-compatible feature set :		مجموعة أعلام الميزات المتوافقة – في وضعية القراءة فقط [27] [72] [85]
				0x0001	RO_COMPAT_SPARSE_SUPER	توصيفات المجموعات ونسخ الكتلة العليا ستكون متناثرة Sparse superblocks (أي ليست في كل مجموعات الكتل) [86]
				0x0002	RO_COMPAT_LARGE_FILE	نظام الملفات يستخدم في تخزين ملفات أكبر من 2 جيجابايت
				0x0004	RO_COMPAT_BTREE_DIR	محتوى الدليل مخزن في شكل شجرة ثنائية أو BTREE ؟! (غير مستخدم في النواة أو حزمة e2fsprogs) أنظر DIR_INDEX
				0x0008	RO_COMPAT_HUGE_FILE	النظام يملك أحجام ملفات تمثل بوحدات من الكتل المنطقية، وليس قطاع 512 بايت، هذا يدل عليه الملف الكبير جدا.
				0x0010	RO_COMPAT_GDT_CSUM	توصيفات المجموعات Group descriptors تملك تدقيق مجاميع [08]
				0x0020	RO_COMPAT_DIR_NLINK	حد الأدلة الثانوية 32,000 في ext3 لم يعد مطبق. و i_links_count في الدليل يعين إلى 1 إذا زاد عن 64,999 [77]
				0x0040	RO_COMPAT_EXTRA_ISIZE	تشير إلى وجود inodes كبيرة على نظام الملفات
				0x0080	RO_COMPAT_HAS_SNAPSHOT	نظام الملفات يملك صورة snapshot [29] (غير مستخدم في النواة / e2fsprogs)
				0x0100	RO_COMPAT_QUOTA	تمكين نظام الحصة (الحصص النسبية للقرص) QUOTA [13]
				0x0200	RO_COMPAT_BIGALLOC	نظام الملفات يدعم bigalloc، هذا يعني تعقب مدبات الملف باستخدام وحدات من العناقيد (من الكتل) بدل الكتل
				0x0400	RO_COMPAT_METADATA_CSUM	دعم تدقيق مجموع البيانات الوصفية.(يقتضي ضمنا GDT_CSUM مع ذلك لا يجب تعيين GDT_CSUM) [88]
				0x0800	RO_COMPAT_REPLICA	نظام الملفات يدعم النسخ طبق الأصل (هذه الميزة ليست في النواة ولا في e2fsprogs)
				0x1000	RO_COMPAT_READONLY	صورة نظام ملفات للقراءة فقط؛ النواة لن تصلها في وضعية القراءة والكتابة ومعظم الأدوات لن تكتب إلى الصورة
				0x2000	RO_COMPAT_ORPHAN_PRESENT	استخدام ملف معزول ! (غير مستخدم في النواة / e2fsprogs)
				0x2000	RO_COMPAT_PROJECT	نظام الملفات يتعقب حصص القرص باستخدام project quotas (هذا نوع جديد من الحصص !) [14]
				0x4000	RO_COMPAT_SHARED_BLOCKS	تمكين مشاركة الكتل في نظام الملفات (في وضعية القراءة فقط!) (مستخدم في e2fsprogs-1.44)
				0x8000	RO_COMPAT_VERITY	تأدية وظيفية تشبه dm-verity لكن على مستوى الملفات للتحقق من صحة الملفات [37] (مستخدم في e2fsprogs-1.44)
s_uuid[16]	0x68 (104)	__u8	16	معرّف وحدة التخزين (بقيمة 128-بت UUID) (كما يظهر في خَرْج blkid ويجب أن يكون فريد)		
s_volume_name[16]	0x78 (120)	char	16	اسم وحدة التخزين Volume label (قيمة 16 بايت، ترميز أسكي / ISO-Latin-1 ينتهي بـ 0) (غير مستخدم تقريبا !)		
s_last_mounted[64]	0x88 (136)	char	64	مسار آخر نقط وصل، أي الدليل أين تم وصل نظام الملفات آخر مرة (هذه قيمة 64 بايت، ترميز أسكي / ISO-Latin-1 تنتهي بـ 0 للتوافق)		
s_algorithm_usage_bitmap	0xC8 (200)	__le32	4	خوارزمية لضغط البيانات. قيمة 32 بت لاختيار نظام ضغط البيانات (غير مستخدم في e2fsprogs / لينكس)		
				0	EXT2_LZV1_ALG	0x00000001	LZV1 (Lev-Zimpel-Vogt)
				1	EXT2_LZRW3A_ALG	0x00000002	LZRW (Lempel-Ziv Ross Williams)
				2	EXT2_GZIP_ALG	0x00000004	GZIP (GNU zip)
				3	EXT2_BZIP2_ALG	0x00000008	BZIP2 (Burrows–Wheeler)
				4	EXT2_LZO_ALG	0x00000010	LZO (Lempel–Ziv–Oberhumer)
تنويه: التخصيص المسبق للدليل ينبغي أن يحدث فقط في حالة تمكين علم EXT4_FEATURE_COMPAT_DIR_PREALLOC						
s_prealloc_blocks	0xCC (204)	__u8	1	التخصيص المسبق للكتل		عدد الكتل المخصص مسبقا.عند إنشاء ملفات اعتيادية [33] (قيمة 8 بت) (غير مستخدم في e2fsprogs / لينكس)

s_prealloc_dir_blocks	0xCD (205)	__u8	1				عدد الكتل المخصص مسبقا للأدلة (قيمة 8 بت) (غير مستخدم في e2fsprogs / لينكس)
s_reserved_gdt_blocks	0xCE (206)	__le16	2				عدد المدخلات المحجوزة GDT من أجل توسع نظام الملفات مستقبلا [50]
دعم نظام ملفات قيد الحوادث سيكون صالح في حالة تعيين EXT4_FEATURE_COMPAT_HAS_JOURNAL							
s_journal_uuid[16]	0xD0 (208)	__u8	16	نظام ملفات قيد الحوادث (jbd2)			معرّف كتلة journal superblock التي تقع بعد superblock في قيد الحوادث الخارجي [114] (قيمة 16 بايت UUID)
s_journal_inum	0xE0 (224)	__le32	4				رقم inode للملف قيد الحوادث journal file (قيمة 32 بت) [20]
s_journal_dev	0xE4 (228)	__le32	4				رقم جهاز للملف قيّد الحوادث journal device في حالة تعيين علم ميزة قيد الحوادث الخارجي (قيمة 32 بت)
s_last_orphan	0xE8 (232)	__le32	4				بداية لائحة البيمة inodes (أو مؤشرات الفهرسة المعزولة) من أجل الحذف [73][132]
s_hash_seed[4]	0xEC (236)	__le32	16				البذرة أو القيمة الابتدائية [24] للهاش باستخدام شجرة HTREE [25] [74]
s_def_hash_version	0xFC (252)	__u8	1				نسخة خوارزمية الهاش الابتدائية المستخدمة في هاش الدليل (فهرسة الأدلة) وستكون إحدى دوال الهاش التشفيرية (قيمة 8 بت) [31] :
				0x00	EXT2_HASH_LEGACY	Legacy	تراثي !
				0x01	EXT2_HASH_HALF_MD4	Half MD4	نصف دالة الهاش التشفيرية إم دي 4
				0x02	EXT2_HASH_TEA	Tea (Tiny Encryption Algorithm)	خوارزمية التشفيرية الصغرى !
				0x03	EXT2_HASH_LEGACY_UNSIGNED	Legacy, unsigned	تراثي!، عدد صحيح لا يحمل إشارة [128]
				0x04	EXT2_HASH_HALF_MD4_UNSIGNED	Half MD4, unsigned	نصف دالة الهاش التشفيرية إم دي 4، لا يحمل إشارة
				0x05	EXT2_HASH_TEA_UNSIGNED	Tea, unsigned	خوارزمية التشفيرية الصغرى!، عدد صحيح لا يحمل إشارة
s_jnl_backup_type	0xFD (253)	__u8	1				نوع النسخة الاحتياطية من قيد الحوادث (المبدئية) journal backup [107]
s_desc_size	0xFE (254)	__le16	2				حجم توصيفات مجموعات الكتل group descriptors، بـ بايتات، في حالة تعيين علم ميزة INCOMPAT_64BIT
s_default_mount_opts	0x100 (256)	__le32	4				خيارات وصل نظام الملفات الابتدائية (قيمة 32 بت) : Default mount options :
				0x0001	EXT4_DEFM_DEBUG		طباعة معلومات التنقيح عند وصل أو إعادة وصل نظام الملفات
				0x0002	EXT4_DEFM_BSDGROUPS		الملفات الجديدة تأخذ معرف مجموعة دليل الاحتواء gid (بدلا من معرف العملية الحالية fsgid)
				0x0004	EXT4_DEFM_XATTR_USER		دعم خصائص ممتدة توفرها مساحة المستخدم
				0x0008	EXT4_DEFM_ACL		دعم قوائم التحكم بالنفاذ ACLs، معيار يونيكس (تصاريح نظام الملفات)
				0x0010	EXT4_DEFM_UID16		لا يدعم UIDs قيم 32-بت
				0x0020	EXT4_DEFM_JMODE_DATA		تنفيذجميع البيانات والبيانات الوصفية إلى قيد الحوادث All data and metadata are committed to the journal
				0x0040	EXT4_DEFM_JMODE_ORDERED		تخليص جميع البيانات Data (من الصوان) إلى القرص قبل تنفيذ البيانات الوصفية Metadata إلى قيد الحوادث Journal
				0x0060	EXT4_DEFM_JMODE_WBACK		ترتيب البيانات غير محفوظ؛ يمكن كتابة البيانات بعد كتابة البيانات الوصفية
				0x0100	EXT4_DEFM_NOBARRIER		تعطيل كتابات الصوان إلى القرص write flushes (راجع آلية حواجز الكتابة [134] BARRIER في EXT4) [123]
				0x0200	EXT4_DEFM_BLOCK_VALIDITY		تعقب كتل البيانات الوصفية في نظام الملفات كي لا تستخدم ككتل بيانات. (هذا الخيار هو في حالة تمكين في 3.18)
				0x0400	EXT4_DEFM_DISCARD		تمكين دعم DISCARD، أين يتم إخبار جهاز التخزين عن الكتل التي أصبحت غير مستخدمة [30] [138]
				0x0800	EXT4_DEFM_NODELALLOC		تعطيل التخصيص المتأخر للكتل[28] (راجع delayed allocation)
s_first_meta_bg	0x104 (260)	__le32	4				هوية أول مجموعة كتل وصفية Meta Block Group في حالة تمكين ميزة meta_bg (قيمة 32 بت) [95]
s_mkfs_time	0x108 (264)	__le32	4				زمن إنشاء نظام الملفات، بالثواني (توقيت يونكس)
s_jnl_blocks[17]	0x10C (268)	__le32	68				نسخة احتياطية من مصفوفة i_block[] journal inode's في 15 عنصر الأولى و i_size_high و i_size في العناصر السادسة والسابعة عشر، على التوالي [107]
دعم 64بت سيكون صالح في حالة تمكين ميزة EXT4_FEATURE_COMPAT_64BIT							
s_blocks_count_hi	0x150 (336)	__le32	4				عدد الكتل الإجمالي (32 بت العليا)
s_r_blocks_count_hi	0x154 (340)	__le32	4				عدد الكتل المحجوزة (32 بت العليا)
s_free_blocks_count_hi	0x158 (344)	__le32	4				عدد الكتل الحرة (32 بت العليا)
s_min_extra_isize	0x15C (348)	__le16	2				جميع inodes يجب أن تملك # بايت على الأقل
s_want_extra_isize	0x15E (350)	__le16	2				inodes الجديدة يجب أن تحجز # بايت
s_flags	0x160 (352)	__le32	4				أعلام نظام ملفات متنوعة Miscellaneous flags :
				0x0001	EXT2_FLAGS_SIGNED_HASH	Signed dirhash in use	قيمة هاش دليل تحمل إشارة في الاستخدام
				0x0002	EXT2_FLAGS_UNSIGNED_HASH	Unsigned dirhash in use	قيمة هاش دليل لا تحمل إشارة في الاستخدام
				0x0004	EXT2_FLAGS_TEST_FILESYS	OK for use on development code	من أجل استخدامها في اختبار شفرة التطوير
				0x0010	EXT2_FLAGS_IS_SNAPSHOT	This is a snapshot image	هذه صورة snapshot (صورة تجميد / استنسخ زمني للحالة الملفات / النظام)
				0x0020	EXT2_FLAGS_FIX_SNAPSHOT	Snapshot inodes corrupted	مؤشرات فهرسة snapshot فاسدة
				0x0040	EXT2_FLAGS_FIX_EXCLUDE	Exclude bitmaps corrupted	مصفوفات ثنائية فاسدة لإقصاء! snapshot (Exclude bitmap) تتعقب الكتل المعينة إلى (snapshot files
s_raid_stride	0x164 (356)	__le16	2				وحدة شريطية في مصفوفة ريد RAID stride [11]
s_mmp_interval	0x166 (358)	__le16	2	منع الوصل المتعدد للنظام الملفات			# عدد ثواني انتظار فحص MMP [90]
s_mmp_block	0x168 (360)	__le64	8				# رقم كتلة بيانات حماية نظام الملفات من الوصل المتعدد MMP
s_raid_stripe_width	0x170 (368)	__le32	4	blocks on all data disks (N*stride)			حجم الشريط في مصفوفة ريد RAID stripe width [12]
s_log_groups_per_flex	0x174 (372)	__u8	1	2 ^ s_log_groups_per_flex			حجم مجموعة الكتل المرنة (عدد مجموعات الكتل التي تشكل مجموعة flex_bg) وسيكون:

s_checksum_type	0x175 (373)	__u8	1	EXT2_CRC32C_CHKSUM	نوع خوارزمية تدقيق مجموع البيانات الوصفية. القيمة الوحيدة الصالحة هي 1 (crc32c)
s_reserved_pad	0x176 (374)	__le16	2		حشو/ محاذاة [98]
s_kbytes_written	0x178 (376)	__le64	8		عدد كيلوبايتات المكتوبة إلى نظام الملفات في فترة حياته (هذا مفيد في حالة تقدير كمية اهتراء خلايا / الكتل على أقراص SSD (نتيجة دورات المسح المحدودة P/E cycles) [138]
s_snapshot_inum	0x180 (384)	__le32	4	صور زمنية انتقائية للنظام snapshot	رقم مؤشر فهرسة الصورة النشيطة snapshot [29] (غير مستخدم في e2fsprogs / لينكس) Inode number of active snapshot
s_snapshot_id	0x184 (388)	__le32	4		هوية تتابعية للصورة النشيطة snapshot (غير مستخدم في e2fsprogs / لينكس) sequential ID of active snapshot
s_snapshot_r_blocks_count	0x188 (392)	__le64	8		عدد الكتل المحجوزة للصورة النشيطة snapshot للاستعمال مستقبلا (غير مستخدم في e2fsprogs / لينكس)
s_snapshot_list	0x190 (400)	__le32	4		رقم مؤشر فهرسة بداية لائحة صور snapshot على القرص. (غير مستخدم في e2fsprogs / لينكس)
s_error_count	0x194 (404)	__le32	4	الأخطاء / مثال: [127]	عدد الأخطاء المنظورة
s_first_error_time	0x198 (408)	__le32	4		زمن وقوع أول خطأ، بعدد الثواني (توقيت يونكس)
s_first_error_ino	0x19C (412)	__le32	4		inode المرتبط بأول خطأ
s_first_error_block	0x1A0 (416)	__le64	8		رقم الكتلة المرتبطة بأول خطأ
s_first_error_func[32]	0x1A8 (424)	__u8	32		اسم الوظيفة أين وقع الخطأ
s_first_error_line	0x1C8 (456)	__le32	4		رقم السطر أين وقع الخطأ
s_last_error_time	0x1CC (460)	__le32	4		زمن أحدث خطأ، بعدد الثواني (توقيت يونكس)
s_last_error_ino	0x1D0 (464)	__le32	4		inode المرتبط بأحدث خطأ
s_last_error_line	0x1D4 (468)	__le32	4		رقم السطر أين وقع أحدث خطأ
s_last_error_block	0x1D8 (472)	__le64	8		رقم الكتلة المرتبطة بأحدث خطأ
s_last_error_func[32]	0x1E0 (480)	__u8	32		اسم الوظيفة أين وقع أحدث خطأ
s_mount_opts[64]	0x200 (512)	__u8	64		سلسلة ASCIIZ (ترميز محارف) من أجل خيارات وصل نظام الملفات [34]
s_usr_quota_inum	0x240 (576)	__le32	4	حصص القرص [13]	رقم مؤشر فهرسة ملف حصة المستخدم — inode number of user quota file
s_grp_quota_inum	0x244 (580)	__le32	4		رقم مؤشر فهرسة ملف حصة المجموعة — inode number of group quota file
s_overhead_blocks	0x248 (584)	__le32	4		العناقيدأو الكتل الفوقانية [15] في نظام الملفات overhead blocks/clusters in fs (هذا الحقل دائما صفر، ويعني أن النواة تقوم بحسابه ديناميكياً)
s_backup_bgs[2]	0x24C (588)	__le32	8		مجموعات الكتل التي تتضمن نسخ من superblock (في حالة تمكين ميزة sparse_super2 [86])
s_encrypt_algos[4]	0x254 (596)	__u8	4		خوارزمية التشفير المستخدمة. (راجع معلومات الموسوعة) التي قد يصل عددها عند الاستخدام إلى 4 خوارزميات في أي وقت؛ شفرات الخوارزميات الصالحة مع أماطها ستكون كالتالي:
				0	EXT4_ENCRYPTION_MODE_INVALID — خوارزمية غير صالحة — حاليا الشفرة تستخدم خوارزمية AES-256-XTS مع محتويات الملفات،
				1	EXT4_ENCRYPTION_MODE_AES_256_XTS — AES إيه إي إس 256-بت في نمط XTS — وتستخدم خوارزمية AES-256-CBC+CTS مع أسماء الملفات [139]
				2	EXT4_ENCRYPTION_MODE_AES_256_GCM — AES إيه إي إس 256-بت في نمط GCM — في inode علم ENCRYPT_FL سيدل على تشفير الكائن (ملف)
				3	EXT4_ENCRYPTION_MODE_AES_256_CBC — AES إيه إي إس 256-بت في نمط CBC
				4	EXT4_ENCRYPTION_MODE_AES_256_CTS — AES إيه إي إس 256-بت في نمط CTS
s_encrypt_pw_salt[16]	0x258 (600)	__u8	16		سولت salt يستخدم من أجل خوارزمية string2key / string-to-key (للتشفير) Salt used for string2key algorithm
s_lpf_ino	0x268 (616)	__le32	4		رقم مؤشر فهرسة الدليل lost+found
s_prj_quota_inum	0x26C (620)	__le32	4		مؤشر الفهرسة inode الذي يتعقب حصص القرص من نوع project quotas [14]
s_checksum_seed	0x270 (624)	__le32	4	crc32c(~0, $orig_fs_uuid)	بذرة [24] تدقيق المجموع المستخدمة في حسابات metadata_csum. هذه القيمة في حالة تعيين csum_seed
s_wtime_hi	0x274 (628)	__u8	1	أختام زمنية	8 بت العليا من حقل زمن الكتابة إلى نظام الملفات آخر مرة s_wtime بعدد الثواني (توقيت يونكس)
s_wtime_hi	0x275 (629)	__u8	1		8 بت العليا من حقل زمن تعديل البيانات آخر مرة s_mtime بعدد الثواني (توقيت يونكس)
s_mkfs_time_hi	0x276 (630)	__u8	1		8 بت العليا من حقل زمن إنشاء نظام الملفات s_mkfs_time بالثواني (توقيت يونكس)
s_lastcheck_hi	0x277 (631)	__u8	1		8 بت العليا من حقل زمن الفحص الأخير s_lastcheck_hi بعدد الثواني (توقيت يونكس)
s_first_error_time_hi	0x278 (632)	__u8	1		8 بت العليا من حقل زمن وقوع أول خطأ s_first_error_time_hi بعدد الثواني (توقيت يونكس)
s_last_error_time_hi	0x279 (633)	__u8	1		8 بت العليا من حقل زمن أحدث خطأ s_last_error_time_hi بعدد الثواني (توقيت يونكس)
s_pad	0x27A (634)	__u8[2]	2		حشو صفر Zero padding
s_reserved[96]	0x27C (636)	__le32	--		حشو إلى نهاية الكتلة [98]
s_checksum	0x3FC (1020)	__le32	4	crc32c(superblock)	تدقيق مجموع Superblock checksum (حساب تدقيق مجموع بنية الكتلة العليا يشمل أيضا FS UUID)

حجم الكتلة الإجمالي 1024 بايت (المصدر: ext4.wiki.kernel.org)

كل مجموعة كتل على نظام الملفات تملك واصف في جدول توصيفات مجموعات كتل [16] وكما تظهر في التخطيط أعلاه، هذه المصفوفة (إن وجدت) ستكون العنصر الثاني داخل مجموعة الكتل. أيضا كل مجموعة كتل في إعداداتها المعمارية تتضمن نسخة كاملة من هذا الجدول ما لم يتم تعيين علم [86] sparse_super. حجم الجدول سيكون وفق عدد مجموعات الكتل. الذي يبدأ عند أول كتلة تتبع superblock (ستكون الكتلة 3 في نظام ملفات كتلة 1k، أو الكتلة 2 في نظام ملفات كتلة 2k أو الأكبر).

لاحظ كيف group descriptor موقع كل من inode table و inode bitmap و block bitmap [32] (أي يمكن أن تطفو) هذا يعني أن superblock و group descriptor table هما فقط من يملك مواقع ثابتة داخل مجموعة الكتل.

آلية flex_bg تستغل هذه الميزة في جمع عدة مجموعات كتل في مجموعة مرنة واحدة ووضع inode tables و inode bitmaps و block bitmaps من كل المجموعات في أول مجموعة من flex group. (راجع flex_bg)

في حال تعيين ميزة meta_bg [95] يتم جمع عدة مجموعات كتل في مجموعة وصفية واحدة. لكن فقط مجموعة الكتل الأولى والثاني والأخيرة في المجموعة الوصفية الأكبر تتضمن واصفات المجموعات داخل meta group. (راجع meta_bg)

لحظ أن الميزتان flex_bg و meta_bg لا تبدوان متنافيتان. (أي يمكن أن تقع في نفس الوقت)

في ext2/3/4 (بدون ميزة 64bit) توصيف مجموعة الكتل group descriptor سيكون بطول 32 بايت فقط وينتهي عند حقل bg_checksum. وعند تمكين ميزة 64bit في ext4، توصيف مجموعة الكتل يتمدد إلى 64 بايت على الأقل؛ هذا الحجم يخزن في superblock. كل block group descriptor سيملك البنية التالية في جدول توصيفات مجموعات كتل block group descriptor table الذي يتضمن مدخلات (معلومات) عن جميع مجموعات الكتل.

block group descriptor في struct ext4_group_desc (لكل مجموعة كتل) (المثال من نظام 32 بت)

```
dd if=/dev/sda1 bs=4096 skip=1 count=1 | dd bs=32 count=1 | hexdump -C
          00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000  6a 02 00 00 7a 02 00 00  8a 02 00 00 6f 2e 56 00  |j...z.......o.V.|
0010  18 02 04 00 00 00 00 00  00 00 00 00 00 00 66 a5  |..............f.|
```

[REMOVED]

رمز تذكري	إزاحة	نوع	حجم			
bg_block_bitmap_lo	0x00 (00)	__le32	4	عنوان الكتلة / موقع المصفوفة الثنائية للكتل (32-بت المنخفضة) Blocks bitmap block		
bg_inode_bitmap_lo	0x04 (04)	__le32	4	عنوان الكتلة / موقع المصفوفة الثنائية للمؤشرات الفهرسة [01] (32-بت المنخفضة) Inodes bitmap block		
bg_inode_table_lo	0x08 (08)	__le32	4	عنوان الكتلة / موقع جدول مؤشرات الفهرسة (32-بت المنخفضة) Inodes table block		
bg_free_blocks_count_lo	0x0C (12)	__le16	2	تعداد الكتل الحرة (16-بت المنخفضة)		
bg_free_inodes_count_lo	0x0E (14)	__le16	2	تعداد مؤشرات الفهرسة الحرة (16-بت المنخفضة)		
bg_used_dirs_count_lo	0x10 (16)	__le16	2	تعداد الأدلة في مجموعة الكتل (16-بت المنخفضة) أي عدد مؤشرات الفهرسة المخصص للأدلة		
bg_flags	0x12 (18)	__le16	2	أعلام مجموعة الكتل	Block group flags :	
				المصفوفة الثنائية وجدول مؤشرات الفهرسة inodes غير مهيئة	EXT4_BG_INODE_UNINIT	0x0001
				المصفوفة الثنائية للكتل غير مهيئة	EXT4_BG_BLOCK_UNINIT	0x0002
				جدول مؤشرات الفهرسة inodes مصفر (ITABLE_ZEROED)	EXT4_BG_INODE_ZEROED	0x0004
bg_exclude_bitmap_lo	0x14 (20)	__le32	4	موقع "Exclude bitmap" لإقصاء! صور snapshot (32-بت المنخفضة) (Exclude bitmap تتعقب الكتل المعينة إلى snapshot files في نظام ملفات Next3)		
bg_block_bitmap_csum_lo	0x18 (24)	__le16	2	تدقيق مجموع المصفوفة الثنائية للكتل (16-بت المنخفضة)	crc32c(s_uuid+grp_num+bbitmap)	
bg_inode_bitmap_csum_lo	0x1A (26)	__le16	2	تدقيق مجموع المصفوفة الثنائية للمؤشرات الفهرسة (16-بت المنخفضة)	crc32c(s_uuid+grp_num+ibitmap)	
bg_itable_unused_lo	0x1C (28)	__le16	2	تعداد مؤشرات الفهرسة الغير مستخدمة (16-بت المنخفضة) [91]		
bg_checksum	0x1E (30)	__le16	2	تدقيق مجموع توصيف المجموعة (أنظر [92]) Group descriptor checksum	crc16(sb_uuid+group+desc)	
الحقول التالية ستكون موجودة فقط في حالة تمكين ميزة 64bit مع حجم s_desc_size أكبر من 32						
bg_block_bitmap_hi	0X20 (32)	__le32	4	موقع المصفوفة الثنائية للكتل (32-بت العليا)		
bg_inode_bitmap_hi	0x24 (36)	__le32	4	موقع المصفوفة الثنائية للمؤشرات الفهرسة (32-بت العليا)		
bg_inode_table_hi	0x28 (40)	__le32	4	موقع جدول inodes (32-بت العليا)		
bg_free_blocks_count_hi	0x2C (44)	__le16	2	تعداد الكتل الحرة (16-بت العليا)		
bg_free_inodes_count_hi	0x2E (46)	__le16	2	تعداد مؤشرات الفهرسة الحرة (16-بت العليا)		
bg_used_dirs_count_hi	0x30 (48)	__le16	2	تعداد الأدلة (16-بت العليا)		
bg_itable_unused_hi	0x32 (50)	__le16	2	تعداد مؤشرات الفهرسة الغير مستخدمة (16-بت العليا)		
bg_exclude_bitmap_hi	0x34 (52)	__le32	4	موقع كتلة "Exclude bitmap" لإقصاء! صور snapshot (32-بت العليا) Exclude bitmap block (المصفوفة الثنائية تتعقب الكتل المعينة إلى snapshot files في Next3)		
bg_block_bitmap_csum_hi	0x38 (56)	__le16	2	تدقيق مجموع المصفوفة الثنائية للكتل (16-بت العليا)	crc32c(s_uuid+grp_num+bbitmap)	
bg_inode_bitmap_csum_hi	0x3A (58)	__le16	2	تدقيق مجموع المصفوفة الثنائية للمؤشرات الفهرسة (16-بت العليا)	crc32c(s_uuid+grp_num+ibitmap)	
bg_reserved	0x3C (60)	__le32	4	حشو إلى بايت 64 [98]		

حجم إجمالي 64 بايت (المصدر: ext4.wiki.kernel.org)

في حالة تعيين ميزة gdt_csum (تدقيق مجموع توصيفات المجموعات) وتعطيل ميزة metadata_csum (تدقيق مجموع البيانات الوصفية) تدقيق مجموع مجموعة الكتل سيكون crc16 بحساب: FS UUID. ورقم المجموعة وبنية توصيف المجموعة.

وفي حالة تعيين ميزة metadata_csum، تدقيق مجموع مجموعة الكتل سيكون 16 بت المنخفضة بحساب تدقيق مجموع: FS UUID. ورقم المجموعة، وبنية توصيف المجموعة.

تدقيق مجموع كل من المصفوفات الثنائية للمؤشرات الفهرسة inode bitmap والمصفوفات الثنائية للكتل block bitmap سيكون بحساب: FS UUID. ورقم المجموعة، وكامل المصفوفة الثنائية bitmap.

المصفوفة الثنائية [32] للكتل البيانات Data Block Bitmap تتعقب استخدام كتل البيانات ضمن مجموعة الكتل. في أنظمة الملفات الصغرى، هذه ستكون بحجم كتلة واحدة، مع موقع غير ثابت. عادة تكون في الكتلة الأولى، أو في الثانية في حالة وجود نسخة superblock في مجموعة الكتل. موقع هذه المصفوفة الثنائية يشر إليه حقل bg_block_bitmap في توصيف المجموعة الخاص Group Descriptor.

المصفوفة الثنائية للمؤشرات الفهرسة Inode Bitmap تعمل بنفس طريقة المصفوفة الثنائية للكتل. غير أن في Inode Bitmap كل بت يمثل inode في جدول Inode Table بدلا من تمثيل كتلة block.

بمعنى آخر inode Bitmap تسجل أية مدخلات في Inode Table قيد الاستخدام. كل مجموعة كتل تملك مصفوفة ثنائية واحدة، و inode Bitmap ستكون بحجم كتلة واحدة، مع موقع غير ثابت. يشير إليه حقل bg_inode_bitmap في توصيف المجموعة الخاص Group Descriptor.

عند إنشاء inode table، بعض المدخلات الأولى في الجدول ستكون محجوزة. (يتم وسمها بالمستعملة) في المراجعة 0 هذه كانت 11 مدخلة، بينما في مراجعة 1 (EXT2_DYNAMIC_REV) واللاحقة عدد مدخلات inodes المحجوزة سوف يحدده حقل s_first_ino في بنية superblock. (راجع "Special inodes").

ملاحظة: في حالة تعيين BLOCK_UNINIT من أجل مجموعة كتل معينة، أجزاء عدة من شفرة النواة وحزمة e2fsprogs [103] ستدعي أن المصفوفة الثنائية للكتل block bitmap تتضمن أصفار (أي،جميع كتل المجموعة حرة) لكن ذلك لا يعني بالضرورة عدم وجود كتل مستخدمة.-

ففي حالة تعيين meta_bg-[95] ستظل bitmaps و group descriptor داخل المجموعة. للأسف دالة () ext2fs_test_block_bitmap2 ستعود بالقيمة '0' لتلك المواقع، وهذا سينتج عنه خَرَج ملتبس في المنقح debugfs.

كما هو حال معظم المصفوفات الثنائية، بت واحد يمثل حالة استعمال كتلة بيانات واحدة أو مدخلة واحدة في Inode Table. هذا يقتضي حجم مجموعة كتل من number_of_bytes_in_a_logical_block * 8.

بالنسبة للكتل كل بت يمثل الوضعة الحالية للكتلة داخل مجموعة الكتل، حيث 1 يعني" كتلة مستعملة" و 0 يعني "كتلة حرة/متوفرة". وأول كتلة من مجموعة الكتل هذه يمثلها بت 0 من بايت 0. والثانية يمثلها بت 1 من بايت 0. والكتلة الثامنة يمثلها بت 7 (البت الأكثر أهمية MSB) من بايت 0 والكتلة التاسعة يمثلها بت 0 (البت الأقل أهمية LSB) من بايت 1 ... إلى آخره.

مثال: جزء من أول مصفوفة ثنائية في أول مجموعة كتل

```
# dumpe2fs /dev/sda1 | grep "Block bitmap"
Block bitmap at [redacted] (+610), Inode bitmap at [redacted] (+634)
(618 x 4096 = 2531328) (634 x 4096 = 2596864) إذن
# dd if=/dev/sda1 bs=4096 skip=618 count=1 | hexdump -Cv
# od -w32 -N 4096 -j 2531328 -A d -t x1 -v group0.dd (file) أو
```

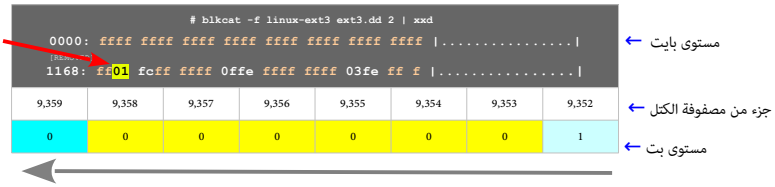
```
# od -w32 -N 4096 -j 2531328 -A d -t x1 -v group0.dd
2531328 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
2531360 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
2531392 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
2531424 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
 [REMOVED]
2532576 ff ff ff ff ff ff ff 07 08 ff 69 00 ff ff ff ff ff 1f 00 00 00 fc ff 7f 00 fe ff ff ff ff ff ff
 [REMOVED]
2535072 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2535104 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2535136 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 [REMOVED]
```

في تمثيل المصفوفة الثنائية للكتل block bitmap غالبا، سيكون هناك 4096 × 8 = 32768 كتلة لكل مجموعة ("Blocks per group" -i | grep -l /dev/sda1 (tune2fs).

وفي تمثيل المصفوفة الثنائية للمؤشرات الفهرسة inode bitmap باعتبار أن حجم inode هو 256 بايت في ext4، وحجم الكتلة 4096 بايت إذن 4096 ÷ 256 = 16 مؤشر فهرسة لكل كتلة

بوجود 32768 بت في كتلة المصفوفة الثنائية (4096 × 8)، نظريا سيكون هناك 32768 مؤشر فهرسة في كل مجموعة، ومن ثم 32768 ÷ 16 = 2048 كتلة في inode table من كل مجموعة.لكن، في الحقيقة، حجم inode table في superblock هو من سيحدد عدد inodes الفعلية لكل مجموعة ("Inodes per group" -i | grep -l /dev/sda1 (tune2fs) مثلا: إذا كان inodes per block group هو 8192 ستكون هناك 512 كتلة في بداية كل مجموعة (8192 ÷ 16 = 512)

مثال آخر عند تحليل مضمون group descriptor نجد أن المصفوفة الثنائية للكتل block bitmap في المجموعة 0 تبدأ عند الكتلة 2. يمكن استخراج مضمون هذه الكتلة باستخدام أداة blkcat [131] أو dd :



قراءة بايتات من اليسار إلى اليمين (على مستوى بايت). بينما القراءة داخل كل بايت من اليمين إلى اليسار (ضمن بايت)

في المثالين السابقين، القيمة " f " تشير إلى تخصيص كتل كثيرة في بداية مجموعة الكتل 0 block group. وفي المثال الثاني، القيمة 0x01 عند بايت 1,169 تقابل الكتل من 9,352 إلى 9,359. في هذا المثال القيمة 0x01 تشير إلى تخصيص الكتلة 9,352، وعدم تخصيص الكتل من 9,353 إلى 9,359.

في نظام ملفات يونكس الاعتيادي ليست مدخلة الدليل [117] ولكن inode[01][116][115]هو من يخزن البيانات الوصفية للملف (أي يخزن الأختام الزمنية، راوبط الكتل، الخصائص الممتدة... إلى آخره، باستثناء اسم ومضمون الملف) ولإيجاد معلومات الملف، يجب التعمق في ملفات الدليل (أي الأدلة) للعثور على مدخلة الدليل المرتبطة بالملف، ثم تحميل inode للحصول على البيانات الوصفية للملف. نظام ملفات (في المراجعة 0.5 واللاحقة) يخزن أيضا نسخة من (حقل) نوع الملف [33] في بنية مدخلة الدليل، (علما أن نوع الملف مخزن أصلا في inode).

كل مجموعة كتل تملك جدول مؤشرات فهرسة واحد Inode Table (يتضمن هياكل من Inodes) بحجم كتلة واحدة أو أكثر. مع موقع غير ثابت، يشير إليه حقل bg_inode_table في توصيف المجموعة الخاص

جدول Inode table عبارة عن مصفوفة خطية من بيانات struct ext4_inode. كتل هذا الجدول تكفي لتخزين على الأقل القيمة: sb.s_inode_size * sb.s_inodes_per_group. رقم مجموعة الكتل التي تتضمن Inode يمكن الحصول عليه بحساب: (inode_number - 1) / sb.s_inodes_per_group. ويمكن الحصول على الإزاحة في جدول المجموعة بحساب: (inode_number - 1) % sb.s_inodes_per_group. علما أن 0 inode لا يستخدم.

تدقيق مجموع مؤشر الفهرسة inode checksum سيكون بحساب كل من: معرف FS UUID. ورقم inode، وبنية inode نفسها.

مدخلة inode table في struct ext4_inode

```
dd if=/dev/sda1 bs=4096 skip=2098698 count=506 | dd bs=256 skip=7105 count=1 | hexdump -Cv
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 b4 81 e8 03 6e 2a 72 00 84 cb c6 5b 5d 95 35 5b |....n*r....[].5[|
0010 39 3d 35 5b 00 00 00 00 e8 03 01 00 18 39 00 00 |9=5[.........9..|
0020 00 00 08 00 01 00 00 00 0a f3 03 00 04 00 00 00 |................|
0030 00 00 00 00 00 00 00 00 02 00 00 00 10 12 00 00 |................|
0040 00 02 00 00 02 00 00 00 48 12 00 00 04 00 00 00 |........H.......|
0050 23 03 00 00 e0 12 00 00 00 00 00 00 00 00 00 00 |#...............|
0060 00 00 00 00 80 54 44 4d 00 00 00 00 00 00 00 00 |.....TDM........|
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0080 20 00 00 00 34 25 12 bc 88 ee 48 29 f0 c7 10 3f |  ...4%....H)...?|
0090 14 3d 35 5b 9c 17 fa 81 00 00 00 00 00 00 00 00 |.=5[............|
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[REMOVED]
```

					حجم / نوع	إزاحة	رمز تذكري
نمط الملف (قيمة 16 بت) هذه ستكون خصائص معيارية للملف في شكل أذون / أنماط / أنواع / أعلام / قيم (قراءة الأنماط في النظام الثماني مثال 0644 أي -rw-r--r-)					2 __le16	0x00 (00)	i_mode
الآخرون يمكنهم التنفيذ / البحث	0x1	S_IXOTH	Others may execute	أذون نفاذ / حقوق النفاذ			
الآخرون يمكنهم الكتابة	0x2	S_IWOTH	Others may write				
الآخرون يمكنهم القراءة	0x4	S_IROTH	Others may read				
أعضاء المجموعة يمكنهم التنفيذ / البحث	0x8	S_IXGRP	Group members may execute				
أعضاء المجموعة يمكنهم الكتابة	0x10	S_IWGRP	Group members may write				
أعضاء المجموعة يمكنهم القراءة	0x20	S_IRGRP	Group members may read				
المالك يمكنه التنفيذ / البحث	0x40	S_IXUSR	Owner may execute				
المالك يمكنه الكتابة	0x80	S_IWUSR	Owner may write				
المالك يمكنه القراءة	0x100	S_IRUSR	Owner may read				
بت تقييد (Sticky bit) ! [17]	0x200	S_ISVTX	Sticky bit	أعلام أنماط خاصة (تنفيذية)			
تعيين معرف المجموعة (GID)	0x400	S_ISGID	Set GID				
تعيين معرف المستخدم (UID)	0x800	S_ISUID	Set UID				
أنبوبة اتصال مسماة	0x1000	S_IFIFO	Named pipe / FIFO	صيغة ملف/ أنواع ملفات متنافية [قيم يجب تعين واحد منها فقط] [33]			
جهاز محرفي	0x2000	S_IFCHR	character special file (character device)				
دليل (مجلد)	0x4000	S_IFDIR	Directory file (Directory)				
جهاز كتل (جهاز كتلي)	0x6000	S_IFBLK	block special file (block device)				
ملف اعتيادي	0x8000	S_IFREG	Regular file				
وصلة رمزية (وصلة لينة) [111]	0xA000	S_IFLNK	Symbolic link, soft link, symlink				
مقبس	0xC000	S_IFSOCK	Unix domain socket				
معرف المالك UID المرتبط بالملف (16-بت المنخفضة)					2 __le16	0x02 (02)	i_uid
حجم الملف بالبايت (32-بت المنخفضة) [75]					4 __le32	0x04 (04)	i_size_lo
زمن النفاذ آخر مرة إلى الملف، بعدد الثواني (توقيت يونكس)	أو سيكون تدقيق مجموع قيمة الخاصية الممتدة في حال تعيين علم EA_INODE			أختام زمنية أو قيمة الخاصية الممتدة [76]	4 __le32	0x08 (08)	i_atime
زمن تغيير آخر مرة مؤشر الفهرسة (نفسه) بعدد الثواني (توقيت يونكس)	أو 32 بت المنخفضة للتعداد المرجعي للقيمة الخاصية في حال تعيين علم EA_INODE				4 __le32	0x0C (12)	i_ctime
زمن تعديل البيانات آخر مرة، بعدد الثواني (توقيت يونكس)	أو رقم مؤشر الفهرسة الذي يمتلك الخاصية الممتدة في حال تعيين علم EA_INODE				4 __le32	0x10 (16)	i_mtime
زمن الحذف، بعدد الثواني (توقيت يونكس). قيمة 32 بت تشير إلى زمن حذف مؤشر الفهرسة					4 __le32	0x14 (20)	i_dtime
معرف المجموعة GID (16-بت المنخفضة)					2 __le16	0x18 (24)	i_gid
تعداد الروابط الصلبة (عدد الروابط إلى مؤشر الفهرسة / الملف) [77] [111]					2 __le16	0x1A (26)	i_links_count
تعداد الكتل"block"-(32-بت المنخفضة) (عدد الكتل المحجوزة للملف. قد تعني حجم الكتلة بالقطاع مثل 512 بايت أو بالكتلة مثل 4096 بايت بتعيين HUGE_FILE) [78]					4 __le32	0x1C (28)	i_blocks_lo
أعلام مؤشرات الفهرسة (قيمة 32 بت تشير إلى كيف سيتصرف تطبيق ext4 عند النفاذ إلى بيانات مؤشر الفهرسة هذا)			Inode flags :		4 __le32	0x20 (32)	i_flags
هذا الملف يستلزم الحذف الأمن "secure deletion" (غير مطبق)	0x00000001	EXT4_SECRM_FL					
يجب حفظ هذا الملف (record for undelete) (غير مطبق)	0x00000002	EXT4_UNRM_FL					

الملف مضغوط (ليس مطبق بالضرورة)		0x00000004	EXT4_COMPR_FL			
كافة الكتابات إلى الملف يجب أن تكون متزامنة (synchronous updates)		0x00000008	EXT4_SYNC_FL			
الملف ثابت، غير قابل للتغيير		0x00000010	EXT4_IMMUTABLE_FL			
يمكن فقط إلحاق الكتابات بالملف (append only)		0x00000020	EXT4_APPEND_FL			
وسيلة dump (1) لا يجب أن تطرح هذا الملف (لا يجب حفظ الملفات عند استخدام dump في نسخ الملفات الاحتياطية)		0x00000040	EXT4_NODUMP_FL			
لا تحدد زمن النفاذ (i_atime)		0x00000080	EXT4_NOATIME_FL			
ملف مضغوط معدل [26] (غير مستخدم) (Dirty compressed file)		0x00000100	EXT4_DIRTY_FL			
الملف يملك عنقود مضغوط أو أكثر (غير مستخدم) (compressed blocks)		0x00000200	EXT4_COMPRBLK_FL			
لا تضغط الملف (النفاذ إلى بيانات مضغوطة خام/أولية) (غير مستخدم)		0x00000400	EXT4_NOCOMPR_FL			
مؤشر فهرسة مشفر. قيمة بت هذه كانت سابقا EXT4_ECOMPR_FL من أجل (خطأ ضغط) والتي لم تستخدم أبدا		0x00000800	EXT4_ENCRYPT_FL			
الدليل يملك فهارس هاش [31] hashed indexes (أي الدليل يستخدم شجرة HTree) (EXT2_BTREE_FL)		0x00001000	EXT4_INDEX_FL			
الدليل السحري magic directory في AFS		0x00002000	EXT4_IMAGIC_FL			
يجب دائما كتابة بيانات الملف من خلال قيد الحوادث [113]		0x00004000	EXT4_JOURNAL_DATA_FL			
لا يجب دمج ذيل الملف File tail (الجزء الأخير من الملف) (غير مستخدم في ext4)		0x00008000	EXT4_NOTAIL_FL			
يجب كتابة جميع بيانات مدخلة الدليل بالتزامن (الأدلة فقط) (راجع dirsync في MOUNT(8))		0x00010000	EXT4_DIRSYNC_FL			
قمة هرميّة الأدلة Top of directory hierarchy		0x00020000	EXT4_TOPDIR_FL			
هذا ملف ضخم huge file (يعين إلى كل ملف ضخم) [54]		0x00040000	EXT4_HUGE_FILE_FL			
مؤشر الفهرسة يستخدم المِدبات		0x00080000	EXT4_EXTENTS_FL			
علم مؤشر الفهرسة في الملفات المحمية بواسطة fs-verity ويسمى Verity protected inode ! (مستخدم في e2fsprogs-1.44)		0x00100000	EXT4_VERITY_FL			
مؤشر الفهرسة يخزن في كتل بياناته قيمة خاصية ممتدة كبيرة large EA		0x00200000	EXT4_EA_INODE_FL			
هذا الملف يملك كتل مخصصة تتجاوز نهاية الملف EOF (مهجور)		0x00400000	EXT4_EOFBLOCKS_FL			
لا تستخدم عملية النسخ عند الكتابة Copy-on-write على الملف Do not cow file (مستخدم في e2fsprogs-1.44 / linux-5.1.7)		0x00800000	FS_NOCOW_FL			
مؤشر فهرسة عبارة عن صورة snapshot [29] (ليس في الخط الرئيسي mainline / مستودع المشروع)		0x01000000	EXT4_SNAPFILE_FL			
حذف الصورة snapshot (ليس في الخط الرئيسي)		0x04000000	EXT4_SNAPFILE_DELETED_FL			
اكتمل تقليص الصورة snapshot (ليس في الخط الرئيسي)		0x08000000	EXT4_SNAPFILE_SHRUNK_FL			
مؤشر فهرسة يملك بيانات مضمنة inline data		0x10000000	EXT4_INLINE_DATA_FL			
إنشاء فروع children بنفس هوية المشروع (ملفات فرعية بنفس هوية الدليل الأم) [14] Create with parents projid		0x20000000	EXT4_PROJINHERIT_FL			
Casefolded file تطبق على الأدلة وأفرعها وتعني أن الدليل يتطلب بحث غير حساسة لحالة الأحرف (مستخدم في e2fsprogs-1.45)		0x40000000	EXT4_CASEFOLD_FL			
محجوز من أجل مكتبة ext4 library		0x80000000	EXT4_RESERVED_FL			
أعلام مرئية للمستخدم-	أعلام مُجمّلة	0x4BDFFF	EXT4_FL_USER_VISIBLE			
أعلام تقبل التعديل من المستخدم [79]		0x4B80FF	EXT4_FL_USER_MODIFIABLE			

مخصص للنظام التشغيل (قيمة 32 بت) (هذا الحقل يملك معاني متعددة وفقا لمنشئ نظام الملفات) Union osd1 (OS dependent 1)						i_osd1
وصف	حجم / نوع	إزاحة	رمز تذكري	وسم		
إصدارة مؤشر الفهرسة Inode version [80] أو 32 بت العليا للتعداد المرجعي للقيمة الخاصية في حالة تعيين EA_INODE	4 __le32	0x00	l_i_version	linux1	4	__le32
"translator" ؟	4 __le32	0x00	h_i_translator	hurd1		
محجوز	4 __le32	0x00	m_i_reserved	masix1		
شجرة مدبات في ext4 (راجع "inode.i_block") أو ستكون مخطط ربط الكتل (أي مؤشرات Pointers إلى الكتل) (15 × "32-بت") [97] [100]					60	i_block[EXT4_N_BLOCKS=15]
إصدارة الملف أو Generation number (مستخدم من قبل بروتوكول نظام الملفات الشبكي NFS) [106]					4	i_generation
كتلة الخاصية الممتدة (32-بت المنخفضة)، و ACLs ستكون إحدى هذه الخصائص الممتدة الكثيرة؛ اسم هذا الحقل ربما كان نتيجة أول استخدام للخصائص الممتدة من أجل ACLs					4	i_file_acl_lo
حجم الدليل/ الملف (32-بت العليا). اسم هذا الحقل في EXT2/3 كان i_dir_acl، رغم ذلك كان يعين إلى الصفر ولم يستخدم أبدا [09] [81]					4	i_size_high / i_dir_acl
عنوان كتلة fragment (هذا الحقل ملغى في ext4)					4	i_obso_faddr
مخصص للنظام التشغيل (قيمة 96 بت) (هذا الحقل يملك معاني متعددة وفقا لمنشئ نظام الملفات) Union osd2 (OS dependent 2)					12	i_osd2
وصف	حجم / نوع	إزاحة	رمز تذكري	وسم		
تعداد الكتل (16-بت العليا) (راجع الملاحظة الملحقة بـ i_blocks_lo) [21]	2 __le16	0x00 (00)	l_i_blocks_high	linux2		
كتلة الخصائص الممتدة (16-بت العليا)، تاريخيا موقع أذون الملف ACL، (راجع الخصائص الممتدة)	2 __le16	0x02 (02)	l_i_file_acl_high			
معرف المستخدم المالك UID ـ(16-بت العليا) [45]	2 __le16	0x04 (04)	l_i_uid_high			
معرف المجموعة GID ـ(16-بت العليا) [45]	2 __le16	0x06 (06)	l_i_gid_high			
تدقيق مجموع مؤشر الفهرسة (16-بت العليا) crc32c(uuid+inum+inode)	2 __le16	0x08 (08)	l_i_checksum_lo			
غير مستخدم	2 __le16	0x0A (10)	l_i_reserved			
محجوز ؟ [21]	2 __le16	0x00 (00)	h_i_reserved1	Hurd2		
نمط الملف (16-بت العليا)	2 __le16	0x02 (02)	h_i_mode_high			

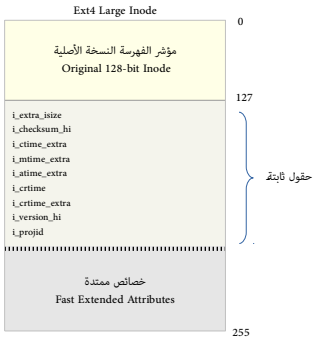
معرف المستخدم المالك UID -(16-بت العليا) [45]	2	__le16	0x04 (04)	h_i_uid_high				
معرف المجموعة GID-(16-بت العليا) [45]	2	__le16	0x06 (06)	h_i_gid_high				
Author code (هوية مؤلّف الملف !) [82]	4	__u32	0x08 (08)	h_i_author				
محجوز [21]	2	__le16	0x00 (00)	h_i_reserved1	masix2			
كتلة الخصائص الممتدة (16-بت العليا)، تاريخيا موقع أذون الملف ACL	2	__le16	0x02 (02)	m_i_file_acl_high				
محجوز	4	__u32	0x04 (04)	m_i_reserved2[2]				
حجم حقول مؤشر الفهرسة الممتدة خلف مؤشر الفهرسة الأصلية ext2، وتشمل هذا الحقل (أنظر للخطاطة أسفل) (Size of this inode - 128)	2	__le16	0x80 (128)	i_extra_isize				
تدقيق مجموع مؤشر الفهرسة (16-بت العليا)	2	crc32c(uuid+inum+inode)	0x82 (130)	i_checksum_hi				
بتات زمن التغيير (حقول Extra) هذا يوفر (فصل زمني) دقة أقل من الثانية sub-second (راجع الأختام الزمنية في inode)	4	__le32	0x84 (132)	i_ctime_extra	أختام زمنية تستخدم نانو ثانية			
بتات زمن التعديل (حقول Extra) هذا يوفر دقة أقل من الثانية	4	__le32	0x88 (136)	i_mtime_extra				
بتات زمن النفاذ للملف (حقول "Extra") هذا يوفر دقة أقل من الثانية	4	__le32	0x8C (140)	i_atime_extra				
زمن إنشاء الملف، بعدد الثواني (توقيت يونكس)	4	__le32	0x90 (144)	i_crtime				
بتات زمن إنشاء الملف (حقول "Extra") هذا يوفر دقة أقل من الثانية	4	__le32	0x94 (148)	i_crtime_extra				
رقم الإصدارة (32-بت العليا) (من أجل نسخة 64 بت)	4	__le32	0x98 (152)	i_version_hi				
هوية المشروع [14] Project ID	4	__le32	0x9C (156)	i_projid				

حجم مؤشر الفهرسة

[Inode Size]

في أنظمة ext2/3، حجم بنية inode [115][116] كان ثابت عند 128 بايت (EXT2_GOOD_OLD_INODE_SIZE) والتسجيلة على القرص كانت بحجم 128 بايت. لكن مع استخدام ext4 صار بالإمكان في زمن التهيئة تخصيص inodes بحجم أكبر، لتمتد المساحة إلى ما وراء النهاية الأصلية في ext2 inode. حجم تسجيلة inode على القرص يسجل في حقل s_inode_size في superblock وعدد بايتات الفعلية المستخدم من struct ext4_inode خلف (128-بايت الأصلية) ext2 inode يسجل في حقل i_extra_isize، هذا سوف يسمح بنمو struct ext4_inode مع النواة الجديدة دون الحاجة إلى ترقية جميع inodes على القرص. الحقول خلف EXT2_GOOD_OLD_INODE_SIZE يجب النفاذ إليها ضمن i_extra_isize.

مبدئيا، تسجيلة ext4 inode بحجم 256 بايت و (اعتبارًا من أكتوبر 2013) بنية inode بحجم 156 بايت (i_extra_isize = 28) المساحة الإضافية بين نهاية بنية inode ونهاية تسجيلة inode يمكن استخدمها لتخزين الخصائص الممتدة.[121] وكل تسجيلة مؤشر الفهرسة يمكن أن تصل إلى حجم كتلة نظام الملفات، مع ذلك هذا ليس له فاعلية كبيرة على النظام.



إيجاد موقع مؤشر الفهرسة

[Finding an Inode]

مؤشرات الفهرسة [115][116] ستكون بترتيب عددي. كل رقم مؤشر فهرسة (مؤشر) في جدول مؤشرات الفهرسة inode table يشير إلى بنية Inode. حجم الجدول يحدد زمن التهيئة لاحتواء أقصى عدد من المدخلات.كل مجموعة كتل تتضمن عدد من مؤشرات الفهرسة Inodes يحدده حقل sb->s_inodes_per_group. في superblock ولأن 0 inode لا يستخدم أصلا، أول Inode في inode table سيكون 1 (وليس 0). يمكن استخدام الصيغة التالية لإيجاد مجموعة الكتل التي يوجد فيها مؤشر الفهرسة:

bg = (inode_num - 1) / sb->s_inodes_per_group.

وللحصول على عنوان بايت (الإزاحة) ضمن inode table، نستخدم: offset = index * sb->s_inode_size.

index = (inode_num - 1) % sb->s_inodes_per_group

	قاعدة	نتيجة	مثال حساب
حجم inode table (بايتات) (الذي يكفي لتخزين على الأقل هذه القيمة):	b.s_inode_size * sb.s_inodes_per_group	= 2072576 بايت	$ echo "8096 * 256" | bc
إزاحة inode ضمن inode table في مجموعة الكتل:	index = (inode_num - 1)% sb -> s_inodes_per_group	= B (11)	$ echo "obase = 16; ibase = 16; (C-1)% 1FA0" | bc
رقم مجموعة الكتل التي تتضمن inode :	bg = (inode_num - 1) / sb -> s_inodes_per_ group.	= مجموعة 0	$ echo "obase = 10; ibase = 16; (C -1) / 1FA0" | bc
عنوان مدخلة inode داخل inode table:	addr = index * sb -> s_inode_size	= 00000b00	$ echo "obase = 16; ibase = 16; B * 100" | bc

يمكنك اختيار : دخل عشري = 10، أو ست عشري 16 في ibase، وخرج عشري 10، أو ست عشري 16 في obase

عينية من حساب مؤشر فهرسة ←

(مثال من ثلاثة مجموعات كتل كانت: 0، 1، 2)

s_inodes_per_group = 1712

رقم فهرسة مؤشر الفهرسة المحلي Local Inode Index	رقم مجموعة الكتل Block Group Number	رقم مؤشر الفهرسة Inode Number
0	0	1
962	0	963
1711	0	1712
0	1	1713
1711	1	3424
0	2	3425

أربعة أختام زمنية مسجلة في 128 بايت المنخفضة من بنية Inode هي زمن تغيير مؤشر الفهرسة ctime (تغيير البيانات الوصفية)، زمن النفاذ (نفاذ للملف أو فتح دليل الملف) atime وزمن تعديل البيانات mtime وزمن الحذف dtime. هذه الحقول الأربعة، أعداد صحيحة 32-بت تحمل إشارة [128] تمثل الثواني منذ توقيت يونكس 1970-01-01 00:00:00 GMT، هذا يعني أن الحقول ستطفو في يناير 2038ـ بالنسبة للمؤشرات الفهرسة التي ليس لها رابط من أي دليل لكنها ما زالت مفتوحة orphan inodes [132] حقل dtime سيفيض لاستخدامه مع لائحة المعزولة orphan list. حقل s_last_orphan في superblock سيشير إلى أول Inode في لائحة؛ حينذاك dtime سيكون رقم مؤشر الفهرسة اليتيم التالي orphaned inode أو يكون صفر إذا لم تكن هناك مؤشرات فهرسة معزولة أخرى orphans.

إذا كان حقل sb->s_inode_size (حجم بنية inode) أكبر من 128 بايت وحقل i_inode_extra يكفي للتطويق i_[cma]time_extra فإن حقول ctime وatime و mtime ستتمدد إلى 64 بت.

في حقل "extra"(32-بت) هذا تستخدم 2 بت المنخفضة لمد حقل الثواني 32-بت إلى 34 بت؛ 30 بت العليا ستوفر ختم زمني بدقة نانو ثانية. وبذلك الأختام الزمنية لن تطفو حتى مايو 2446، و dtime لم يمدد.

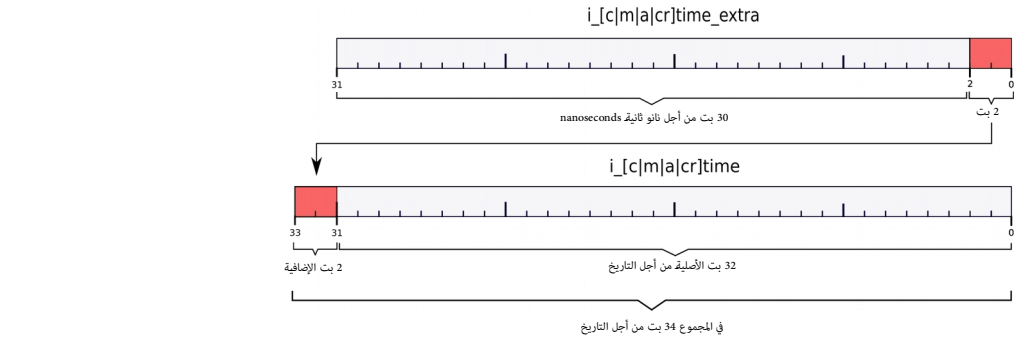
في ext4 هناك أيضا حقل خامس crtime في الأختام الزمنية يسجل زمن إنشاء inode؛ هذا الحقل بعرض 64-بت ومترجم بنفس أسلوب i_[cma]time_extra.

حقول crtime و dtime لن تكون متاحة من خلال الواجهة الاعتيادية ()stat، رغم أن المنقح debugfs سيعلن عنهم (راجع أيضا bug-findutils / coreutils patches) أمثلة:

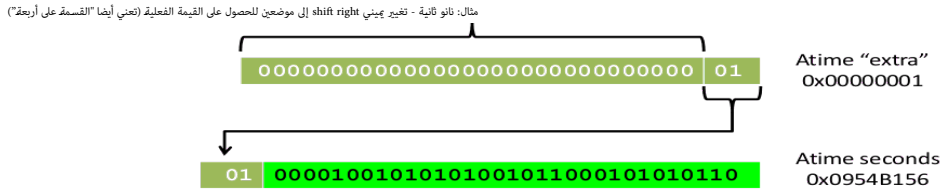
```
debugfs: stat /
ctime: 0x5a2520d3:c43db60c -- Mon Dec  4 10:17:55 2017
atime: 0x5a2520fb:98591bd8 -- Mon Dec  4 10:18:35 2017
mtime: 0x5a2520d3:c43db60c -- Mon Dec  4 10:17:55 2017
crtime: 0x59f78ef9:00000000 -- Mon Oct 30 20:43:37 2017
```

```
debugfs: stat giis.txt
ctime: 0x4805d3eb -- Wed Apr 16 15:54:43 2008
atime: 0x4805d3e7 -- Wed Apr 16 15:54:39 2008
mtime: 0x4805d3eb -- Wed Apr 16 15:54:43 2008
dtime: 0x4805d445 -- Wed Apr 16 15:56:13 2008
```

```
# stat testfile
Access: 2017-08-22 15:10:27.131878596 +0200
Modify: 2017-08-22 15:06:45.229604104 +0200
Change: 2017-08-22 15:06:45.229604104 +0200
 Birth: -
```



مثال: نانو ثانية - تغيير يميني shift right إلى موضعين للحصول على القيمة الفعلية (تعني أيضا "القسمة على أربعة")



تستخدم قيمة للزمن 32-بت مع إشارة إضافة إلى (2^{32} × (بتات الإضافية في توقيت يونكس)). بعبارة أخرى:

مثال ترجمة القيمة العشرية عبر طرفية لينكس	نطاق زمني صالح valid time range	tv_sec-64-بت مترجمة / مفككة الترميز Decoded 64-bit tv_sec	تعديل 32-بت مع إشارة إلى 64-بت tv_sec Adjustment for signed 32-bit to 64-bit tv_sec	بت الأكثر أهمية في توقيت 32-بت MSB of 32-bit time	بتات توقيت يونكس إضافية Extra epoch bits
date -d "@-2147483648"	13-12-1901 إلى 13-12-1969 (سالبة)	-0x80000000 - -0x00000001	0x100000000	1	0 0
date -d @2147483647	01-01-1970 إلى 19-01-2038	0x000000000 - 0x07fffffff	0	0	0 0
	19-01-2038 إلى 07-02-2106	0x080000000 - 0x0ffffffff	0x100000000	1	0 1
	07-02-2106 إلى 25-02-2174	0x100000000 - 0x17fffffff	0x100000000	0	0 1
	25-02-2174 إلى 16-03-2242	0x180000000 - 0x1ffffffff	0x200000000	1	1 0
	16-03-2242 إلى 04-04-2310	0x200000000 - 0x27fffffff	0x200000000	0	1 0
	04-04-2310 إلى 22-04-2378	0x280000000 - 0x2ffffffff	0x300000000	1	1 1
	22-04-2378 إلى 10-05-2446	0x300000000 - 0x37fffffff	0x300000000	0	1 1

في 06:28:15 UTC، الأحد 7 فبراير 2106، توقيت يونكس سوف يصل إلى 4,294,967,295 ثانية، هذا يعني أن في الأنظمة التي تحفظ الوقت بأعداد صحيحة موجبة 32-بت لا تحمل إشارة، سيكون ذلك التاريخ أقصى ما يمكن تحقيقه. ويعني أيضا في تلك الأنظمة، أن الثانية التالية ستترجم بشكل خاطئ إلى 00:00:00 الثلاثاء 1 يناير 1970 UTC.

في 15:30:08 UTC، الأحد 4 ديسمبر 292,277,026,596 (لكن أين ستكون البشرية في هذا التاريخ!)، نسخ 64-بت من ختم يونكس الزمني ستتوقف عن العمل، لأنها ستطفو أكبر قيمة يمكن حفظها في رقم 64-بت لا يحمل إشارة. هذا تقريبا 22 مرة عمر الكون المقدر حاليا، والذي هو 1.37×10^{10} مليار سنة. هناك أيضا أخطاء قديمة في ترميز وفك ترميز التواريخ بعد 2038، ويبدوا أنها لم تعرف الحل حتى إصدار نواة 3.12 وحزمة e2fsprogs 1.42.8 [103] أيضا نسخ سابقة من أنوية 64-بت تستخدم بالخطأ بتات توقيت يونكس الإضافية 1,1 في التواريخ بين 1901 و 1970، غالبا في مرحلة ما سيتم إصلاح النواة، وأداة e2fsck ستعمل على إصلاح هذا الوضع، على افتراض أن تعمل قبل 2310ـ

مضمون i_block في مؤشر الفهرسة

inode.i_block

60 بايت في inode.i_block يمكن استخدامها بطرق مختلفة، وفقا لنوع الملف [33] الذي يصفه inode عموما، الملفات والأدلة العادية تستخدمها لفهرسة كتل الملفات، والملفات الخاصة تستخدمها لأغراضها الخاصة.

وصلات رمزية / وصلات لينة

Symbolic Links

هدف الوصلة الرمزية [111] سوف يخزن في هذا الحقل إذا كان طول سلسلة الهدف (النصية) أقل من 60 بايت. ما عدا ذلك تستخدم المديات أو [122] ربط الكتل في تخصيص كتل البيانات لتخزين هدف الرابط (الملف).

العنونة المباشرة والغير مباشرة للكتل

Direct/Indirect Block Addressing

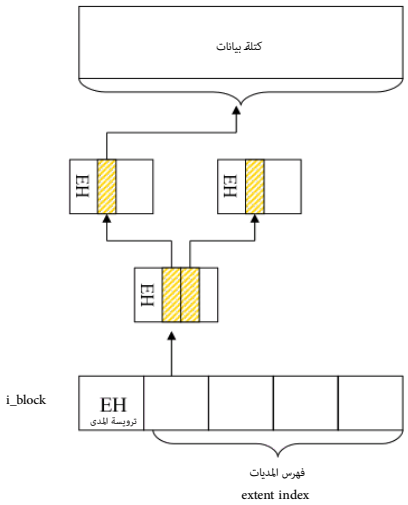
في أنظمة ملفات ext2/3، أرقام كتل الملف تعين إلى أرقام الكتل المنطقية عبر ربط للكتل 1-1 (قد يصل إلى) ثلاث مستويات. لإيجاد الكتلة المنطقية التي تخزن كتلة الملف، الشفرة ستبحث في كامل هذه البنية المعقدة على نحو متزايد [97] لاحظ هنا، لا يوجد رقم سحري (توقيع) أو تدقيق مجموع يضمن خلاء الكتلة من ما يسمى القمامة garbage.

<table>
<tr><td>تشير إلى</td><td>إزاحة i.i_block</td></tr>
<tr><td>تعين (أو ربط) مباشر إلى كتل الملف من 0 إلى 11 (أي أول 12 كتلة تعمل مباشرة من داخل inode)</td><td>0 إلى 11</td></tr>
<tr><td>كتلة غير مباشرة : (كتل الملف من 12 إلى (4 / $block_size) + 11 أو من 12 إلى 1035 في حالة كتل 4 كيلوبايت أي block_size = 4096)</td><td rowspan="3">12</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة</td></tr>
<tr><td>تعين مباشر إلى كتل (4 / $block_size) (1024 في حالة كتل 4 كيلوبايت)</td><td>من 0 إلى (4 / $block_size)</td></tr>
<tr><td>كتلة غير مباشرة مزدوجة: (كتل الملف من 12 + 4/$block_size إلى (4 / $block_size) ^ 11 + (4 / $block_size) + $_2) أي من 1036 إلى 1049611 في حالة كتل 4 كيلوبايت)</td><td rowspan="5">13</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة مزدوجة</td></tr>
<tr><td>تعين إلى الكتل الغير مباشرة (4 / $block_size) (1024 في حالة كتل 4 كيلوبايت)</td><td rowspan="3">من 0 إلى (4 / $block_size)</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة</td></tr>
<tr><td>تعين مباشرة إلى كتل (4 / $block_size) (1024 في حالة كتل 4 كيلوبايت)</td><td>من 0 إلى (4 / $block_size)</td></tr>
<tr><td>كتلة غير مباشرة ثلاثية: (كتل الملف من 4 / $block_size) ^ 12 + (4 / $block_size) + 2 $) إلى 12 + (4 / $block_size) + 2 ^ (4 / $block_size) + 3 ^ (4 / $block_size) أي من 1049612 إلى 1074791436 في حالة كتل 4 كيلوبايت)</td><td rowspan="7">14</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة ثلاثية</td></tr>
<tr><td>تعين إلى الكتل الغير مباشرة المزدوجة (4 / $block_size) (1024 في حالة كتل 4 كيلوبايت)</td><td rowspan="5">من 0 إلى (4 / $block_size)</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة مزدوجة</td></tr>
<tr><td>تعين إلى الكتل الغير مباشرة (4 / $block_size) (1024 في حالة كتل 4 كيلوبايت)</td><td rowspan="3">من 0 إلى (4 / $block_size)</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة</td></tr>
<tr><td>تعين إلى الكتل (4 / $block_size) (1024 في حالة كتل 4 كيلوبايت)</td><td>من 0 إلى (4 / $block_size)</td></tr>
</table>

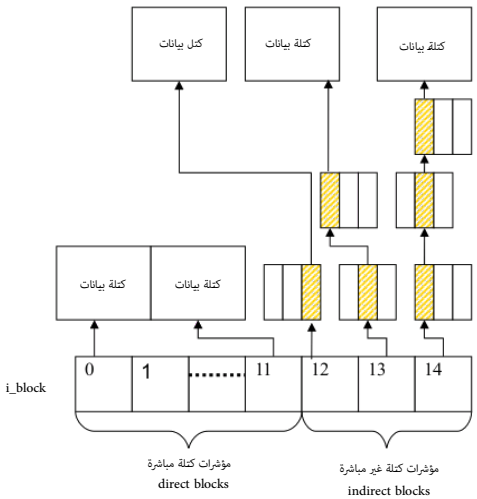
* في هذه الصيغ القسمة كانت على 4 لأن كل رقم كتلة منطقية مخزن في 4 بايت.

لاحظ مع مخطط ربط الكتل هذا، سيكون من الضروري ملء الكثير من بيانات الربط [122] حتى مع الملف الكبير لملتمس! وهذا هو السبب في إنشاء مخطط ربط المديات [120] Extents، (الموصوف أدناه).

لاحظ أيضا أن الملف الذي يستخدم مخطط الربط هذا mapping scheme لا يمكن أن يوضع أعلى من 2^{32} كتلة[100][97]

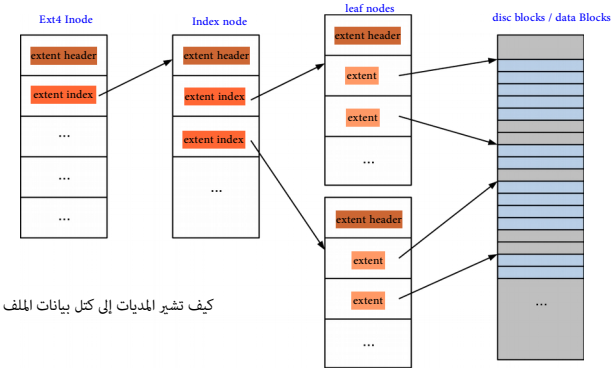


ربط (شجرة) المديات
extent map



ربط الكتلة/الكتل الغير مباشر(ة)
indirect block map

في ext4، الربط بين الكتل المنطقية والملف استبدل بشجرة مديات [120][100] (المدى: سلسلة كتل فيزيائية متماسة، يمكنها ربط ما يصل إلى 128 ميغابايت من المساحة باستخدام كتلة 4 كيلوبايت) في المخطط القديم، كان تخصيص رتل متماس من 1,000 كتلة يستلزم كتلة غير مباشرة لربط جميع المدخلات الألف؛ باستخدام المديات انخفض الربط إلى مدى واحد struct ext4_extent حيث ee_len = 1000. (الكتل التي يغطيها المدى) في حال تمكين ميزة flex_bg. يمكن تخصيص ملفات كبيرة جدا بمدى واحد. وسينتج عن ذلك خفض كبير في استخدام كتل البيانات الوصفية وتحسن في فاعلية القرص. لكن هذه الميزة تستلزم تعيين علم المديات (0x80000) في Inode.



كيف تشير المديات إلى كتل بيانات الملف

عموما، بنية المدى تنقسم إلى ثلاثة:

- ترويسة مدى Extent header
- مؤشر / فهرس مدى Extent index (عقد داخلية)
- مدى extent (عقد طرفية)

المثال على اليسار للمؤشر فهرسة ملف يملك 60 بايت لتخزين بنية مديات الملف، حجم مدخلة المدى الواحدة extent entry هو 12 بايت، وبالتالي، يمكن تخزين 5 مدخلات مدى (أو بالضبط 4) مباشرة في inode. وكل بنية مدى تبدأ بترويسة مدى extent header بحجم 12 بايت،

راجع أكثر تفاصيل بنية ترويسة المدى في الجدول التالي.

ترويسة المدى extent header متبوعة بعقد المدى extent nodes، التي يمكن أن تكون عقد داخلية inner nodes من شجرة المدى extent tree [فهارس المدى extent indexes] وتشير إلى ترويسات المدى الأخرى، أو تكون عقد طرفية leaves [مديات extents] تشير إلى كتل بيانات data block runs.

راجع بنية هذه المدخلات في الجداول التالية.

المديات Extents مرتبة في شكل شجرة. كل عقدة في الشجرة تبدأ مع ترويسة struct ext4_extent_header.إن كانت العقدة عقدة داخلية interior node (أي عقدة تملك عقدة ابن واحدة على الأقل) (eh.eh_depth > 0). الترويسة يتبعها تجسدات eh.eh_entries من struct ext4_extent_idx؛ كل واحدة من مدخلات الفهرس هذه [40] تشير إلى كتلة تتضمن عقد أكثر في شجرة المديات. وإن كانت العقدة عقدة طرفية leaf node (ليس لها عقد أبناء وترتبط بعقدة واحدة أخرى فقط) (eh.eh_depth == 0) حينذاك الترويسة يتبعها تجسدات eh.eh_entries من struct ext4_extent؛ هذه التجسدات تشير إلى كتل البيانات الخاصة بالملف.

العقدة الجذرية root node (عادة، تكون أعلى عقدة) في شجرة المديات تخزن في inode.i_block,التي تسمح بتسجيل (تخزين) المديات الأربعة الأولى بدون استخدام كتل بيانات وصفية إضافية.(أي لا تفهرس في شجرة)

ترويسة المديات Ext4 Extent header مسجلة في بنية struct ext4_extent_header، (بطول 12 بايت) في شجرة extent tree

```
dd if=/dev/sda1 skip=2098698 bs=4096 count=506 | dd skip=6113 bs=256 count=1 | hexdump -Cv
      00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000  b4 81 e8 03 0e 30 a9 00  2e ae db 5b 2e ae db 5b  |.....0.....[...[|
0010  d3 18 33 5b 00 00 00 00  e8 03 01 00 a8 54 00 00  |..3[.........T..|
0020  00 00 08 00 01 00 00 00  0a f3 01 00 04 00 01 00  |................|
0030  00 00 00 00 00 00 00 00  5f 89 20 00 00 00 1c 00  |........_. .....|
0040  02 00 00 00 1e 00 00 00  c1 0c 14 00 20 00 00 00  |............ ...|
0050  60 00 00 00 85 0f 14 00  80 00 00 00 80 01 00 00  |`...............|
0060  b3 98 14 00 20 d6 a1 2b  00 00 00 00 00 00 00 00  |.... ..+........|
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0080  20 00 00 00 98 e4 08 6f  30 b7 3d 2a 90 24 4b 7e  | ......o0.=*.$K~|
0090  e2 17 33 5b f4 05 4f 14  00 00 00 00 00 00 00 00  |..3[..O.........|
[REMOVED]
```

رقم سحري (التوقيع) 0xF30A		Magic number, 0xF30A	2	_le16	حجم / نوع	0x00 (00)	إزاحة	eh_magic	رمز تذكري
عدد مدخلات المدى الصالحة التي بعد الترويسة			2	_le16		0x02 (02)		eh_entries	
العدد الأقصى للمدخلات المدى بعد الترويسة			2	_le16		0x04 (04)		eh_max	
عمق عقدة المدى هذه في شجرة المديات [101]		4*(((blocksize - 12)/12)^n) >= 2^32 is 5	2	_le16		0x06 (06)		eh_depth	
رقم توليد الشجرة (هذا يستخدمه نظام الملفات الموزع المتوازي، لوستر Lustre، وليس نظام الملفات المعياري ext4)			4	_le32		0x08 (08)		eh_generation	

مدخلة المدى من أجل العقد الداخلية في شجرة المديات، تعرف أيضا باسم index nodes، ومسجلة في بنية struct ext4_extent_idx، (بطول 12 بايت)

عقدة الفهرسة index node هذه تغطي كتل الملف من الكتلة 'block' فصاعدا (أين نجد المديات تحت هذه العقدة في الشجرة)		رقم الكتلة المنطقية	4	_le32	حجم / نوع	0x00 (00)	إزاحة	ei_block	رمز تذكري
رقم كتلة عقدة المدى (32-بت المنخفضة) في المستوى التالي المنخفض في الشجرة.عقد الشجرة المشار إليها قد تكون عقدة داخلية أخرى أو عقدة طرفية	عنوان الكتلة الفيزيائية		4	_le32		0x04 (04)		ei_leaf_lo	
16-بت العليا من الحقل السابق			2	_le16		0x08 (08)		ei_leaf_hi	
غير مستخدم !			2	_le16		0x0A (10)		ei_unused	

مدخلة المدى من أجل العقد الطرفية في شجرة المديات مسجلة في بنية struct ext4_extent، (بطول 12 بايت)

```
# dd if=/dev/sda1 skip=1572896 bs=4096 count=506 | dd skip=741 bs=256 count=1 | hexdump -Cv
      00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0500  b4 81 e8 03 13 00 00 00  a6 e0 d7 5b a6 e0 d7 5b  |...........[...[|
0510  a6 e0 d7 5b 00 00 00 00  e8 03 01 00 08 00 00 00  |...[............|
0520  00 00 08 00 01 00 00 00  0a f3 01 00 04 00 00 00  |................|
0530  00 00 00 00 00 00 00 00  01 00 00 00 05 83 18 00  |................|
[REMOVED]
```

رقم كتلة الملف الأولى التي يغطيها هذا المدى (مكان هذا المدى النسبي إلى بداية الملف) [115]		رقم الكتلة المنطقية	4	_le32	حجم / نوع	0x00 (00)	إزاحة	ee_block	رمز تذكري
عدد الكتل التي يغطيها هذا المدى [102]			2	_le16		0x04 (04)		ee_len	
رقم الكتلة (16-بت عليا) التي يشير لها هذا المدى	رقم الكتلة الفيزيائية من أول كتلة في المدى، أي البداية الفعلية للمدى على القرص	عنوان الكتلة الفيزيائية	2	_le16		0x06 (06)		ee_start_hi	
رقم الكتلة (32-بت المنخفضة) التي يشير لها هذا المدى			4	_le32		0x08 (08)		ee_start_lo	

بنية struct ext4_extent_tail (بطول 4 بايت)			حجم / نوع	إزاحة	رمز تذكري
تدقيق مجموع كتلة المدى extent block	crc32c(uuid+inum+igeneration+extentblock)	4	_le32	0x00 (00)	eb_checksum

قبل تقديم تدقيق مجاميع البيانات الوصفية، ترويسة المدى+مدخلات المدى كانت تترك 4 بايت على الأقل بدون تخصيص في نهاية كل كتلة شجرة مديات لذلك وضع تدقيق مجموع 32-بت في هذه المساحة. و 4 مديات في inode لا تحتاج تدقيق مجموع لأن تدقيق مجموع inode محسوب. تدقيق المجموع سيكون بحساب: FS UUID، ورقم inode، وتوليد inode وكامل كتلة المدى حتى حقل تدقيق المجموع (دون حسابه).

البيانات المضمنة / البيانات المباشرة — Inline Data

يمكنك تخزين أول 60 بايت من بيانات الملف هنا في حال تمكين ميزة (INCOMPAT_INLINE_DATA) في نظام الملفات (Super Block) وتعيين علم مؤشر الفهرسة (EXT4_INLINE_DATA_FL).

مدخلات الدليل — Directory Entries

الأدلة تستخدم من أجل تنظيم الملفات بشكل مرتبي/ شجري. حيث كل دليل يمكن أن يتضمن أدلة أخرى، أو ملفات اعتيادية، إلى آخره. الأدلة نفسها تخزن في كتل بيانات يشير إليها Inode. ويمكن تمييزها بنوع الملف S_IFDIR المخزن في حقل i_mode.

المدخلة الثانية في Inode table تتضمن Inode يشير إلى بيانات الدليل الجذر Root directory ؛ المحدد بالثابت EXT4_ROOT_INO. في أنظمة يونكس وشبه يونكس (مثل لينكس، ومينكس) الملفات تظهر دائما تحت الدليل الجذر حتى وإن كانت مخزنة على أجهزة فيزيائية مختلفة. وعند إنشاء أي دليل سيبدأ دائما بمدخلتين "." و ".." (المخفية) حتى وإن كان الدليل فارغ. [49]. في المراجعة 0 الأدلة يمكن تخزينها فقط في قائمة متصلة linked list directory، وفي المراجعات اللاحقة تستخدم الأدلة المفهرسة Indexed Directory. هذه الأخيرة ستكون متوافقة خلفيا [99] مع أدلة القائمة المتصلة (الخطية) ؛ وسيتحقق ذلك بإدراج تسجيلات فهارس الهاش hash indexes.

في ext4، الدليل تقريبا ملف مسطح (من مدخلات الدليل) [117]: يربط سلسلة بايت اختيارية (عادة، بترميز أسكي) برقم مؤشر فهرسة inode number على نظام الملفات هذا الأخير يمكن أن يملك مدخلات دليل كثيرة تشير إلى نفس رقم Inode، وتعرف باسم الروابط الصلبة [111] ولأنها كذلك لا يمكنها الإشارة إلى الملفات على أنظمة الملفات الأخرى وبذلك مدخلات الدليل يمكن إيجادها بقراءة كتلة / كتل البيانات المرتبطة بملف الدليل (أي الدليل) للمدخلة الدليل المحددة المطلوبة

الأدلة (التقليدية) الخطية (القائمة المتصلة) — Linear (Classic) Directories

مبدئيا، كل دليل يسرد مدخلاته في مصفوفة خطية تقريباً لكن ليس بالمعني الموجود في الذاكرة. لأن مدخلات الدليل ليست مجزأة على كتل نظام الملفات. ومن ثم، القول الأصح الدليل سلسلة من كتل البيانات. كل كتلة تتضمن مصفوفة خطية من مدخلات الدليل. نهاية كل مصفوفة مدخلات في مصفوفة الكتل مدلول عليه ببلوغ نهاية الكتلة؛ المدخلة الأخيرة في الكتلة تملك طول تسجيلة يحتل كامل الكتلة حتى نهايتها. طبعا نهاية كامل الدليل مدلول عليه ببلوغ نهاية الملف. مدخلات الدليل الغير مستخدمة مدلول عليها بمؤشر الفهرسة = 0 .(أنظر للعنية [119]) نظام الملفات مبدئيا يستخدم بنية struct ext4_dir_entry_2 من أجل مدخلات الدليل أو يستخدم struct ext4_dir_entry في حالة تعطيل علم ميزة "filetype".

صيغة مدخلة الدليل الأصلية ستكون struct ext4_dir_entry، وبطول تقريبا 263 بايت، مع ذلك، للتأكد تحتاج الرجوع إلى dirent.rec_len على القرص.

بنية مدخلة الدليل struct ext4_dir_entry، (الصيغة الأصلية)	حجم	نوع	إزاحة	رمز تذكري
رقم inode الذي تشير إليه مدخلة الدليل هذه	4	__le32	0x00 (00)	inode
طول مدخلة الدليل هذه (يجب أن يكون من مضاعفات العدد 4)	2	__le16	0x04 (04)	rec_len
طول اسم الملف	2	__le16	0x06 (06)	name_len
اسم الملف	255	char	0x08 (08)	name[EXT4_NAME_LEN]

بما أن أسماء الملفات لن تكون أطول من 255 بايت. صيغة مدخلة الدليل الحديثة (المراجعة 0 من ext2) اقتطعت 8 بت العليا من حقل name_len لتخزين علم نوع الملف [33]، ربما لتجنب تحميل كل Inode أثناء البحث المتعمق في شجرة الدليل directory tree. هذه الصيغة ستكون ext4_dir_entry_2 وبطول تقريبا 263 بايت. مع ذلك، للتأكد تحتاج الرجوع إلى dirent.rec_len. على القرص

بنية مدخلة الدليل ext4_dir_entry_2 (الصيغة الحديثة)

```
debugfs -R "cat <518977>" /dev/sda1 | hexdump -C
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 41 eb 07 00 0c 00 01 02 2e 00 00 00 08 ee 05 00 |A...............|
0010 0c 00 02 02 2e 2e 00 00 42 eb 07 00 0c 00 04 01 |........B.......|
0020 74 68 69 73 cc ee 07 00 0c 00 02 01 69 73 00 00 |this........is..|
0030 cd ee 07 00 0c D8 01 01 61 00 00 00 33 ef 07 00 |........a...3...|
0040 10 00 06 01 73 69 6d 70 6c 65 00 00 56 ef 07 00 |....simple..V...|
0050 b4 0f 09 01 64 69 72 65 63 74 6f 72 79 00 00 00 |....directory...|
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
(REMOVED)
```

رقم inode الذي تشير إليه مدخلة الدليل هذه [10]	حجم	نوع	إزاحة	رمز تذكري
رقم inode الذي تشير إليه مدخلة الدليل هذه [10]	4	__le32	0x00 (00)	Inode
طول مدخلة الدليل هذه [10]	2	__le16	0x04 (04)	rec_len
طول اسم الملف [10]	1	__u8	0x06 (06)	name_len
شفرة نوع الملف [33] [10] (تستخدم فقط 3 بت المنخفضة، بقية بتات محجوزة الآن) وستكون إحدى هذه: File type code :	1	__u8	0x07 (07)	file_type
اسم الملف [10]	255	char	0x08 (08)	name[EXT4_NAME_LEN]

نوع ملف مجهول	Unknown	Unknown	0x00
ملف اعتيادي	EXT4_FT_REG_FILE	Regular file	0x01
ملف دليل	EXT4_FT_DIR	Directory file (Directory)	0x02
ملف جهاز محرفي	EXT4_FT_CHRDEV	character special file (character device file)	0x03
ملف جهاز كتل	EXT4_FT_BLKDEV	block special file (block device file)	0x04
ملف صوان / أنبوبة اتصال مسماة	EXT4_FT_FIFO	Named pipe / FIFO (Buffer File)	0x05
ملف مقبس (IPC socket)	EXT4_FT_SOCK	Unix domain socket	0x06
وصلة رمزية (وصلة لينة) [111]	EXT4_FT_SYMLINK	Symbolic link, soft link, symlink	0x07

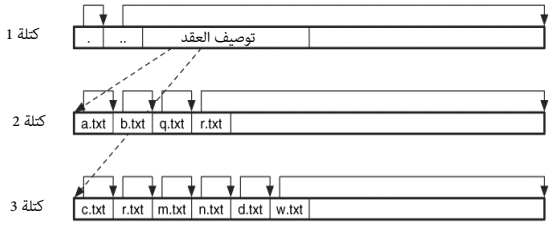
بنية ذيل مدخلة الدليل struct ext4_dir_entry_tail		حجم	نوع	إزاحة	رمز تذكري
رقم inode (يجب أن يكون صفر)		4	__le32	0x00 (00)	det_reserved_zero1
طول مدخلة الدليل هذه، (يجب أن يكون 12)		2	__le16	0x04 (04)	det_rec_len
طول اسم الملف، (يجب أن يكون صفر)		1	__u8	0x06 (06)	det_reserved_zero2
نوع الملف (زائف) (يجب أن يكون 0xDE)		1	__u8	0x07 (07)	det_reserved_ft
تدقيق مجموع الكتلة الطرفية للدليل Directory leaf block	crc32c(uuid+inum+dirblock)	4	__le32	0x08 (08)	det_checksum

لإضافة تدقيق المجموع إلى كل دليل الكلاسيكية هذه، وضعت المدخلة الزائفة struct ext4_dir_entry في نهاية كل كتلة طرفية لحفظ تدقيق المجموع. طول مدخلة الدليل سيكون 12 بايت. مع تعيين حقل رقم مؤشر الفهرسة وحقل name_len إلى الصفر لخداع الرمجيات القديمة وجعلها تتجاهل مدخلة الدليل التي تبدو شاغرة. وتدقيق المجموع سيخزن في مكان الاسم الاعتيادي.تدقيق مجموع الكتلة الطرفية للدليل سيكون بحساب: FS UUID، ورقم inode للدليل، ورقم توليد inode للدليل، وكامل كتلة مدخلة الدليل إلى مدخلة الدليل المزيفة (بدون حسابها)

المصفوفة الخطية من مدخلات الدليل لم تكن ذات فاعلية ولذلك أضيفت في ext3، ميزة Htree [25] التي توفر شجرة بحث ثنائية متزنة ذاتيا وسريعة مع مفاتيح (مستخرجة) من (قيمة) هاش [31] اسم مدخلة الدليل (الملفات مرتبة على أساس الهاش من اسم الملف وليس الاسم) في حالة تعيين علم (0x1000) EXT4_INDEX_FL في Inode. الدليل المعني سوف يستخدم شجرة HTree (الدليل المفهرس) [42] في تنظيم وإيجاد مدخلات الدليل. وللتوافق خلفيا [99] في وضعية القراءة فقط مع ext2، هذه الشجرة ستكون مخفية داخل ملف الدليل (أي دليل)؛ ومقنعة في صفة كتل بيانات دليل " فارغة"!.

كما ذكرنا سابقا، نهاية جدول مدخلات الدليل الخطية مدلول عليه بمدخلة تشير إلى Inode 0 ؛ هذا يستخدم لخداع خوارزمية المسح الخطي القديمة كي تظن أن بقية كتلة الدليل فارغة ومن ثم تتجاوزها. كل كتلة في الدليل تشير إلى عقدة في الشجرة، جذر الشجرة دائما في أول كتلة بيانات في الدليل. (العقدة الوحيدة في الطبقة العليا) وكما في ext3، المدخلات '..' و '.' يجب أن تظهر في بداية هذه الكتلة الأولى، لذلك وضعت هنا في شكل اثنان من struct ext4_dir_entry_2s. ولم تخزن في الشجرة. بقية عقدة الجذر root node (أي الكتلة) تتضمن بيانات وصفية عن الشجرة وتوصيف العقد المتضمن الربط [122] بين قيمة الهاش وعنوان الكتلة hash->block map لإيجاد العقد المنخفضة في Htree.

نظام التشغيل يستخدم توصيفات العقد node descriptors في تحديد الكتلة التي يقفز إليها من أجل قيمة الهاش في الخطاطة التالية تظهر عدة ملفات في طرفيتين. أول كتلة تتضمن ترويسة وتوصيف عقد والثانية والثالثة تتضمن مدخلات دليل ملفات.



هذا الدليل يملك hash trees وطرفيتين two leaves. الشجرة تستخدم مدخلات الدليل directory entries لذلك يمكن معاملتها كالدليل العادي.

كل كتلة في الدليل تشير إلى عقدة في الشجرة، (كل عقدة تتضمن ملفات تملك قيمة هاش متسلسلة) العقد التي ليست طرفية تملك هياكل بيانات تشير إلى الطبقة التالية، وهناك طبقتان في الدليل الأصغر، وأول كتلة ستكون العقدة الوحيدة في الطبقة العليا. عدد طبقات العقد يمكن أن يصل إلى ثلاث في شجرة فهرس الهاش hash index tree. لمعرفة الكتل التي تشير إلى العقدة في الطبقة التالية ستكون هناك هياكل بيانات للتوصيف العقد لكن قبل ذلك ستكون ترويسة، تبدأ بعد مدخلة الدليل ".." حقول ترويسة توصيف العقد node descriptor header تظهر في الجدول أسفل.

إذا كان حقل dx_root.info.indirect_levels بقيمة غير الصفر شجرة HTree سوف تملك مستويين؛ كتلة البيانات التي يشير لها الربط في عقدة الجذر root node's map ستكون عقدة داخلية interior node، وستكون مفهرسة [40] بواسطة قيمة الهاش الدنيا minor hash. العقد الداخلية Interior nodes في هذه الشجرة تتضمن struct ext4_dir_entry_2 ملء بالأصفار متبوعة بالربط بين قيم الهاش الدنيا وعناوين الكتل minor_hash->block map لإيجاد العقد الطرفية leaf nodes. والعقد الطرفية تتضمن مصفوفة خطية linear array من جميع struct ext4_dir_entry_2 ؛ جميع هذه المدخلات (من المفترض) أن [31] تهاش نفس القيمة hash to the same value. إن كان هناك فيض overflow، المدخلات ببساطة تفيض في العقدة الطرفية التالية leaf node، ويتم تعيين بت LSB من hash (في ربط العقدة الداخلية interior node map) التي أوصلتنا إلى العقدة الطرفية التالية هذه.

للبحث المتعمق في الدليل وفق شجرة HTree، الشفرة تحسب hash من اسم الملف المطلوب وتستخدمها لإيجاد رقم الكتلة المقابلة. إذا كانت الشجرة مسطحة، الكتلة ستكون مصفوفة خطية من مدخلات الدليل التي يمكن سبرها ؛ ما عدا ذلك، يتم حساب قيمة الهاش الدنيا minor hash من اسم الملف وتستخدم مقابل هذه الكتلة لإيجاد رقم الكتلة الثانية الثالثة المقابل، رقم الكتلة الثالثة هذه ستكون مصفوفة خطية من مدخلات الدليل.

للبحث المتعمق في الدليل كمصفوفة خطية (كما تفعل الشفرة القديمة)، الشفرة ببساطة تقرأ كل كتلة بيانات في الدليل. الكتل المستخدمة من أجل HTree ستبدو بدون مدخلات (بجانب '.' و '..') لذلك فقط العقد الطرفية leaf nodes ستبدو بمضمون مهم.

جذر شجرة htree في struct dx_root، (بطول كتلة بيانات كاملة) [43]

```
debugfs -R "cat <123456>" /dev/sda1 | hexdump -C
     00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 44 fb 05 00 0c 00 01 02  2e 00 00 00 76 00 00 00 |D...........v...|
0010 f4 0f 02 02 2e 2e 00 00  00 00 00 00 01 08 00 00 |................|
0020 fc 01 10 00 01 00 00 00  16 88 a0 0d 10 00 00 00 |................|
0030 fa 19 a1 1a 08 00 00 00  1a 34 db 2b 0c 00 00 00 |.........4.+....|
0040 c4 fa 9a 3c 04 00 00 00  7a fe 30 4b 0d 00 00 00 |...<....z.0K....|
0050 9a 61 6c 5d 07 00 00 00  92 d5 6d 6b 0f 00 00 00 |.al]......mk....|
0060 f8 fa 4f 78 02 00 00 00  8a 8a 4f 89 0a 00 00 00 |..Ox......O.....|
0070 26 f4 13 9c 05 00 00 00  30 08 7b ae 09 00 00 00 |&.......0.{.....|
0080 58 00 52 c2 03 00 00 00  92 2f 94 d1 0b 00 00 00 |X.R....../......|
0090 7e 7c 74 e2 06 00 00 00  0e 62 7d f0 0e 00 00 00 |~|t.....b}.....|
[REMOVED]
```

رقم مؤشر فهرسة هذا الدليل [130]	this directory "."	Linked Directory Entry (fake_dirent 1)	ext4_directory_dx_dot_entry مدخلتين من نوع دليل قائمة متصلة (للتوافق خلفيا) [99] [117]	4	__le32	0x00 (00)	dot.inode
طول هذه التسجيلة، 12				2	__le16	0x04 (04)	dot.rec_len
طول هذا الاسم، 1				1	u8	0x06 (06)	dot.name_len
نوع ملف هذه المدخلة (0x02 = دليل) (في حالة تعيين علم الميزة)	EXT2_FT_DIR=2			1	u8	0x07 (07)	dot.file_type
	".\0\0\0"			4	char	0x08 (08)	dot.name[4]
رقم مؤشر فهرسة الدليل الأم	parent directory ".."	Linked Directory Entry (fake_dirent 2)		4	__le32	0x0C (12)	dotdot.inode
حجم_الكتلة - 12، طول التسجيلة يكفي لتغطية جميع بيانات Htree	block_size - 12			2	__le16	0x10 (16)	dotdot.rec_len
طول الاسم، 2				1	u8	0x12 (18)	dotdot.name_len
نوع ملف هذه المدخلة (0x02 = دليل) (في حالة تعيين علم الميزة)				1	u8	0x13 (19)	dotdot.file_type
	"..\0\0"			4	char	0x14 (20)	dotdot_name[4]
صفر			ext4_directory_dx_root_info بنية معلومات جذر الدليل المفهرس Indexed Directory Root Information ترويسة توصيف عقدة شجرة الهاش hash tree node descriptor header	4	__le32	0x18 (24)	struct dx_root_info reserved_zero
إصدارة الهاش، هذه 8 بت تمثل نسخة الهاش المستخدمة في الدليل المفهرس وستكون إحدى هذه :				1	u8	0x1C (28)	struct dx_root_info hash_version

تراثي !	DX_HASH_LEGACY	Legacy	0x00
نصف دالة الهاش التشفيرية إم دي 4	DX_HASH_HALF_MD4	Half MD4	0x01
خوارزمية التشفيرية الصغرة !Tiny Encryption Algorithm	DX_HASH_TEA	Tea	0x02
تراثي!، (عدد صحيح) لا يحمل إشارة	DX_HASH_LEGACY_UNSIGNED	Legacy, unsigned	0x03
نصف دالة الهاش التشفيرية إم دي 4، لا يحمل إشارة	DX_HASH_HALF_MD4_UNSIGNED	Half MD4, unsigned	0x04
خوارزمية التشفيرية الصغرة!، لا يحمل إشارة	DX_HASH_TEA_UNSIGNED	Tea, unsigned	0x05

طول معلومات الشجرة، هذه 8 بت تمثل طول بنية معلومات الدليل المفهرس (dx_root) ؛ حاليا تساوي 0x08					struct dx_root_info.info_length
عمق Htree، لا يمكن أن يكون > 3 إذا تم تعيين INCOMPAT_LARGEDIR؛ ما عدا ذلك، لا يمكن أن يكون > 2 [83]					struct dx_root_info.indirect_levels
محجوز					struct dx_root_info.unused_flags
العدد الأقصى من مدخلات الدليل المفهرس dx_entries التي يمكن أن تتبع هذه الترويسة، زائد 1 من أجل الترويسة نفسها	مدخلة توصيف العقدة الأولى first node descriptor entry [118] [43] (حالة خاصة)	2	__le16	0x20 (32)	limit
العدد الفعلي من مدخلات الدليل المفهرس dx_entries التي تتبع هذه الترويسة، زائد 1 من أجل الترويسة نفسها		2	__le16	0x22 (34)	count
رقم الكتلة (داخل ملف الدليل) الذي يقابل hash=0		4	__le32	0x24 (36)	block
أكبر عدد ممكن من مدخلات struct dx_entry قيم 8-بت يمكن أن يتناسب مع بقية حجم كتلة البيانات	ext4_directory_dx_entry entries[]	--	struct dx_entry	0x28 (40)	Entries[0]

العقدة الداخلية في Htree مسجلة في بنية struct dx_node، (بطول كتلة بيانات كاملة)			حجم / نوع	إزاحة	رمز تذكري
صفر، لجعلها تبدو مثل هذه المدخلة غير مستخدمة	مزيفة! / صفرية zeroed out	ext4_fake_directory_entry (Interior node)	4	0x00 (00)	fake.inode
حجم الكتلة، من أجل إخفاء جميع بيانات dx_node			2	0x04 (04)	fake.rec_len
صفر، لا يوجد اسم لمدخلة الدليل هذه "الغير مستخدمة" "unused"			1	0x06 (06)	name_len
صفر، لا يوجد نوع ملف لمدخلة الدليل هذه "الغير مستخدمة" "unused"			1	0x07 (07)	file_type
العدد الأقصى من مدخلات الدليل المفهرس dx_entries التي يمكن أن تتبع هذه الترويسة، زائد 1 من أجل الترويسة نفسها	مخفية!		2	0x08 (08)	limit
العدد الفعلي من مدخلات الدليل المفهرس dx_entries التي تتبع هذه الترويسة، زائد 1 من أجل الترويسة نفسها			2	0x0A (10)	count
رقم الكتلة (داخل ملف الدليل) الذي يقابل قيمة هاش الأدنى من هذه الكتلة. هذه القيمة مخزنة في الكتلة الأم			4	0x0E (14)	Block
أكبر عدد ممكن من مدخلات struct dx_entry قيم 8-بت يمكن أن يتناسب مع بقية حجم كتلة البيانات		ext4_directory_dx_entry entries[]	--	0x12 (18)	Entries[0]

ربط الهاش[31] hash maps في struct dx_root و struct dx_node ومسجل في مدخلة الدليل المفهرس struct dx_entry، (بطول 8 بايت)			حجم / نوع	إزاحة	رمز تذكري
شفرة هاش [31] (32 بت هاش من اسم الملف المعلن من قبل هذه المدخلة) Hash code	مدخلات توصيف عقدة شجرة الهاش hash tree node descriptor entries		4	0x00 (00)	Hash
رقم كتلة العقدة التالية في HTree (داخل ملف الدليل، وليس كتل نظام الملفات) Block number			4	0x04 (04)	Block

بنية تدقيق المجموع dx_tail (بطول 8 بايت) ستكون في نهاية كل كتلة htree block			حجم / نوع	إزاحة	رمز تذكري
(zero right now)			u32	0x00 (00)	dt_reserved
تدقيق مجموع كتلة دليل HTree	crc32c(uuid+inum+dxblock)		__le32	0x04 (04)	dt_checksum

في حالة تمكين تدقيق مجاميع البيانات الوصفية، 8 بايت الأخيرة من كتلة الدليل (بطول واحدة من dx_entry) تستخدم في تخزين بنية struct dx_tail، التي تتضمن تدقيق المجموع.

في هياكل dx_root/dx_node مدخلات limit و count تضبط عند الضرورة كي تتناسب dx_tail داخل الكتلة. إذا لم تكن هناك مساحة من أجل مدخلة dx_tail يبلغ المستخدم بتنفيذ e2fsck -D لإعادة بناء فهرسة الدليل directory index (التي ستتأكد من وجود مساحة من أجل تدقيق المجموع). تدقيق المجموع سيكون بحساب: معرف FS UUID، وترويسة فهرس HTree index (dx_root أو dx_node) وجميع indices HTree (dx_entry) المستخدمة، وكتلة الذيل (dx_tail).

الخصائص الممتدة

Extended Attributes

معظم خصائص الملفات، مثل ملفات الأدلة، وصلات الرمزية، ملفات الأجهزة...إلى آخره تقع في inode المرتبط بالملف. بعض الخصائص الأخرى متوفرة فقط كخصائص ممتدة [121] للملفات الاعتيادية والأدلة، هذه الخصائص الممتدة هي امتدادات للخصائص العادية المرتبطة بكافة مؤشرات الفهرسة inodes في النظام. وتستخدم غالبا في توفير تأدية وظيفية إضافية في نظام الملفات – ميزات أمان إضافية - مثل قوائم التحكم بالنفاذ ACLs التي يمكن تطبيقها كخصائص ممتدة. وهذه سيكون النفاذ إليها بصفتها كائنات ذرية [46] القراءة تجلب كامل قيمة الخاصية وتخزنها في الصوان. والكتابة تستبدل أية قيمة سابقة بقيمة جديدة.

الخصائص الممتدة (xattrs) تخزن في كتلة بيانات مستقلة على القرص تشير إليها حقول inode.i_file_acl* من inodes ويبدو أن أول استخدام للخصائص الممتدة كان من أجل تخزين أذون الملفات ACLs وبيانات الأمن الأخرى (selinux). كما يمكن للمستخدمين مع خيار الوصل user_xattr تخزين الخصائص الممتدة طالما أن جميع أسماء الخصائص تبدأ باسم التصنيف "user"؛ لكن هذا التقييد يبدو أنه اختفى بإصدار لينكس 3.0 (النواة).

والخصائص الممتدة يمكن أن نجدها في مكانين. الأول بين نهاية وبداية كل مدخلة inode. مثلا إذا كان inode.i_extra_isize = 28 و sb.inode_size = 256، إذن 256 - (128 + 28) = 100 بايت ستكون متوفرة لتخزين الخصائص الممتدة في inode.

المكان الثاني الذي يمكن أن نجد فيه الخصائص الممتدة هو الكتلة التي يشير لها inode.i_file_acl.لكن منذ إصدار لينكس 3.11، لا يمكن لهذه الكتلة أن تتضمن مؤشر إلى كتلة خصائص ممتدة ثانية (أو حتى بقية الكتل في العنقود) نظريا، يمكن تخزين كل قيمة خاصة في كتلة بيانات مستقلة، مع ذلك منذ لينكس 3.11 الشفرة لا تسمح بذلك. عموما، المفاتيح (Attribute names) يفترض أن تكون سلاسل ASCIIZ، في شكل <namespace>.<attribute name> بينما القيم يمكن أن تكون سلاسل أو بيانات ثنائية.

الخصائص الممتدة، عندما تخزن بعد inode تملك هذه الترويسة ext4_xattr_ibody_header (بطول 4 بايت)			حجم / نوع	إزاحة	رمز تذكري
رقم سحري (التوقيع) من أجل التعريف، (مشغل لينكس يعين هذه القيمة، لكن، يبدو أن e2fsprogs لا تتفحص هذه القيمة ؟)	0xEA020000		__le32	0x00 (00)	h_magic

بداية كتلة الخصائص الممتدة struct ext4_xattr_header (بطول 32 بايت)

```
# blkcat -f ext3 ext3-2.dd 1238
0000 00 00 02 ea 01 00 00 00 01 00 00 00 74 47 05 e8 |............tG..|
0016 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0032 06 03 a8 e9 51 47 00 00 00 00 19 00 00 00 a8 e9 51 47 |....QG..........QG|
0048 73 6f 75 72 63 65 00 00 00 02 dc 03 00 00 00 00 |source..........|
0064 24 00 00 00 25 00 ad 01 00 00 00 00 00 00 00 00 |$...%...........|
```

[REMOVED]			حجم / نوع	إزاحة	رمز تذكري
رقم سحري (التوقيع) من أجل التعريف (32 بت)	EXT4_XATTR_MAGIC = 0xEA020000		__le32	0x00 (00)	h_magic
تَعْداد المراجع [38] (عدد الملفات التي تستخدم هذه الكتلة / الملفات التي تملك نفس الخصائص الممتدة)			__le32	0x04 (04)	h_refcount
عدد كتل القرص المستخدمة من قبل الخصائص الممتدة [44]			__le32	0x08 (08)	h_blocks
قيمة هاش hash value (32 بت) [31] جميع الخصائص (كي يستطيع نظام التشغيل بسهولة تحديد ما إذا كان الملفان يملكان نفس الخصائص)			__le32	0x0C (12)	h_hash
تدقيق مجموع كتلة الخصائص الممتدة (FS UUID، رقم كتلة 64-بت لكتلة الخصائص الممتدة، وكامل الكتلة (الترويسة+ المدخلات))	crc32c(uuid+id+xattrs)		__le32	0x10 (16)	h_checksum
(zero right now)			__u32	0x14 (20)	h_reserved[2]

بنية struct ext4_xattr_header أو struct ext4_xattr_ibody_header سوف تتبعها مصفوفة من مدخلات struct ext4_xattr_entry: كل واحدة بطول 16 بايت على الأقل. عندما تخزن في كتلة خارجية، مدخلات struct ext4_xattr_entry يجب أن تكون بترتيب مفروز sorted order بهذا الشكل: e_name_index، e_name_len ثم وأخيرا e_name الخصائص المخزنة داخل Inode لا تحتاج أن تكون مخزنة بترتيب مفروز.

بنية مدخلة الخصائص (مدخلات أسماء الخصائص الممتدة) struct ext4_xattr_entry:

```
# blkcat -f ext3 ext3-2.dd 1238
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 00 00 02 ea 01 00 00 00 01 00 00 00 74 47 05 e8 |............tG..|
0016 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0032 06 01 c0 03 00 00 00 00 19 00 00 00 a8 e9 51 47 |..............QG|
0048 73 6f 75 72 63 65 00 00 00 02 dc 03 00 00 00 00 |source..........|
0064 24 00 00 00 25 00 ad 01 00 00 00 00 00 00 00 00 |$...%...........|
[REMOVED]
0944 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0960 77 77 77 2e 64 69 67 74 69 74 61 6c 2d 65 76 69 |www.digtital-evi|
0976 64 65 6e 63 65 2e 6f 72 67 00 00 00 01 00 00 00 |dence.org.......|
0992 a1 00 06 00 02 00 04 00 4a 00 00 00 02 00 06 00 |........J.......|
1008 f4 01 00 00 04 00 04 00 10 00 06 00 20 00 04 00 |............ ...|
[REMOVED]
```

رمز تذكري	إزاحة	حجم / نوع		[REMOVED]
e_name_len	0x00 (00)	__u8	1	طول الاسم (8 بت لا تحمل إشارة) (طول الاسم يحدد طول المدخلة، والمدخلة التالية تبدأ عند حدود 4 بايت التالية)
e_name_index	0x01 (01)	__u8	1	(رقم) فهرس اسم الخاصية (نوع الخصائص) [110] (8 بت لا تحمل إشارة) نوع المدخلة يمكن أن يكون بإحدى القيم التالية: Attribute name index :

(رقم) فهرس الاسم	ثابت	سابقة المفتاح (مساحات الأسماء)	ملاحظة
0	--	NULL	لا توجد سابقة
1	EXT4_XATTR_INDEX_USER	"user." (User space attribute)	لا تملك قيود بخصوص التسمية أو المحتويات
2	EXT4_XATTR_INDEX_POSIX_ACL_ACCESS	"system.posix_acl_access" (any file or directory)	هذه تستخدمها النواة من أجل ACLs
3	EXT4_XATTR_INDEX_POSIX_ACL_DEFAULT	"system.posix_acl_default" (directories only)	
4	EXT4_XATTR_INDEX_TRUSTED	"trusted." (Trusted space attribute)	تستخدمها فقط النواة !
5	EXT4_XATTR_INDEX_LUSTRE	LUSTRE (not currently used by Linux)	حاليا غير مستخدم في لينكس!
6	EXT4_XATTR_INDEX_SECURITY	"security." (Security space attribute)	تستخدمها SELinux
7	EXT4_XATTR_INDEX_SYSTEM	"system."	فقط inline_data
8	EXT4_XATTR_INDEX_RICHACL	"system.richacl"	فقط أنوية SuSE
9	EXT4_XATTR_INDEX_ENCRYPTION		تشفير ؟ Encryption
10	EXT4_XATTR_INDEX_HURD	(راجع summerofcode)	محجوز من أجل Hurd

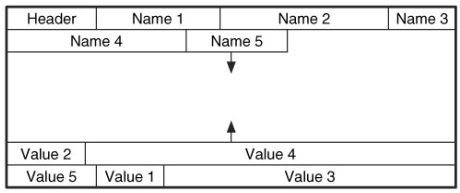
e_value_offs	0x02 (02)	__le16	2	موقع قيمة هذه الخاصية على كتلة القرص حيث تخزن (أي إزاحة بايت في الكتلة المحددة) [108] (16 بت لا تحمل إشارة)
e_value_inum	0x04 (04)	__le32	4	مؤشر الفهرسة الذي يخزن القيمة. الرقم 0 يشير إلى أن القيمة في نفس الكتلة مثل هذه المدخلة. هذا الحقل يستخدم فقط في حالة تمكين ميزة INCOMPAT_EA_INODE
e_value_size	0x08 (08)	__le32	4	حجم قيمة الخاصية (32 بت لا تحمل إشارة) (عدد بايتات في القيمة)
e_hash	0x0C (12)	__le32	4	قيمة الهاش [31] الخاصة بقيمة الخاصية واسم الخاصية [109] hash value of name and value
e_name[e_name_len]	0x10 (16)	char	16	اسم الخاصية attribute name (بترميز أسكي) لا يشمل المحرف الصفري (رمز لاشيء) في الذيل ! trailing NULL

قيم الخصائص Attribute values يمكنها أن تتبع نهاية جدول المدخلات و المحاذاة يجب أن تكون على حدود 4 بايت (أي المدخلة التالية تبدأ عند حدود 4 بايت التالية) القيم تخزن بداية من نهاية الكتلة وتنمو باتجاه جدول xattr_header/xattr_entry.عندما يصطدم الاثنان، الفيض يوضع في كتلة مستقلة على القرص. إذا امتلئت كتلة القرص، نظام الملفات يعود بالخطأ -ENOSPC.

الحقول الأربعة الأولى من ext4_xattr_entry تعين إلى الصفر للدلالة على نهاية لائحة المفاتيح key list

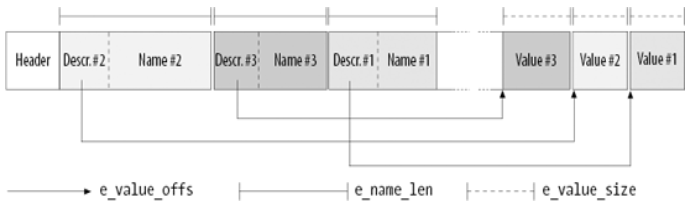
ترويسة كتلة خصائص ممتدة	Attribute Block Header
مدخلة الخاصية 1	Attribute Entry 1
مدخلة الخاصية 2	Attribute Entry 2
مدخلة الخاصية 3	Attribute Entry 3
4 بايت صفرية	4 null bytes
مساحة بدون استخدام	unused space...
قيمة الخاصية 1	Attribute Value 1
قيمة الخاصية 3	Attribute Value 3
قيمة الخاصية 2	Attribute Value 2

النمو نزولا



النمو صعودا

كتلة الخصائص الممتدة: تملك ثلاثة أقسام. أول 32 بت تستخدمها الترويسة header، القسم الثاني يتضمن لائحة بمدخلات أسماء الخصائص (entry descriptors)، القسم الثالث يبدأ من نهاية الكتلة ويتجه للأعلى، ويتضمن القيم لكل زوج خصائص attribute pairs، التي قد لا تكون في نفس الترتيب مثل مدخلات الأسماء. يمكن رؤية ذلك في الشكلين على اليمين واليسار أعلاه. كما يمكنك طرح جميع الخصائص الممتدة للملف بالأمر getfattr -d -m – file



تخطيط كتلة تتضمن خصائص ممتدة

فهارس أسماء الخصائص

Attribute Name Indices

منطقيا، الخصائص الممتدة عبارة عن تسلسل من أزواج قيمة = مفتاح- key=value أو name:value المفاتيح (أسماء الخصائص) ترتبط بشكل دائم بالملفات والأدلة، ويفترض أن تكون سلاسل منتهية بصفر.تشبه السلاسل البيئة المرتبطة بالعملية. والخاصة قد تكون محددة أو غير محددة. وإن كانت محددة، قيمتها يمكن أن تكون خالية أو غير خالية. ولخفض المساحة التي تستهلكها المفاتيح على القرص، ستقارن بداية سلسلة المفتاح [40] بفهرس اسم الخاصية. إذا وجد تطابق، يتم تعيين حقل (رقم) فهرس اسم الخاصية، مع إزالة سلسلة التقابل من اسم المفتاح. الجدول التالي يعرض تعيين قيم فهارس الأسماء إلى سوابق المفاتيح.

(رقم) فهرس الاسم	سابقة المفتاح (مساحات الأسماء)	ملاحظة
0	NULL	لا توجد سابقة
1	"user."	من أجل تسجيل الخواص المحددة من التطبيقات / هذه بدون قيود على التسمية والمضمون (ملفات وأدلة)
2	"system.posix_acl_access"	تستخدمها النواة من أجل ACLs
3	"system.posix_acl_default"	
4	"trusted."	من أجل تسجيل خواص ينبغي للنواة فقط النفاذ إليها (لا ينبغي للعمليات العادية النفاذ إليها باستثناء العمليات التي تملك capability (CAP_SYS_ADMIN
5	LUSTRE (not currently used by Linux)	حاليا غير مستخدم في لينكس!
6	"security."	من أجل تسجيل الخواص الأمنية للملف / تستخدمها SELinux و capabilities (محجوزة من أجل وحدات أمان نواة لينكس (Kernel security modules
7	"system."	من أجل تسجيل الخواص الأخرى المرتبطة بالنظام، يمكن أن يتحكم فيها مالك الملف (حاليا فقط inline_data) (في الأصل تستخدمها النواة من أجل POSIX ACLs)
8	"system.richacl"	فقط أنوية SuSE
9		تشفير ؟ Encryption
10	(راجع summerofcode)	محجوز من أجل Hurd

المستخدم يستطيع إنشاء أي زوج (بأداة setfattr) في هذه الحالة، user سيكون مساحة الاسم، مثال، إذا كان مفتاح الخاصية "user.fubar" (اسم الخاصية + اسم التصنيف والنقطة) يعين (رقم) فهرس اسم الخاصية إلى 1 ويسجل اسم "fubar" على القرص.

قوائم التحكم بالنفاذ (معيار بوزيكس)

POSIX ACLs

قوائم التحكم بالنفاذ[121] معيار بوزيكس ACLs POSIX تخزن في نسخة مصغرة من صيغة ACL الداخلية (وحزمة libacl's) في نواة لينكس.الاختلاف الرئيسي سيكون في رقم النسخة المختلف (1) وحقل e_id يخزن فقط من أجل أذون ACLs المستخدم والمجموعة المسماة named user, named group. جزء من قيمة POSIX ACL attribute تبدأ ببنية بيانات الترويسة (حقل واحد) ثم لائحة من المدخلات (بيانات الجدول التالي من ملف ext4_acl.h).

بنية بيانات الترويسة posix_acl_xattr_header + بنية بيانات المدخلات posix_acl_xattr_entry

```
# blkcat -f ext3 ext3-2.dd 1238
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 00 00 02 ea 01 00 00 00 01 00 00 00 74 47 05 e8 |............tG..|
0016 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0032 06 01 c0 03 00 00 00 00 19 00 00 00 a8 e9 51 47 |..............QG|
0048 73 6f 75 72 63 65 00 00 00 02 dc 03 00 00 00 00 |source..........|
0064 24 00 00 00 25 00 ad 01 00 00 00 00 00 00 00 00 |$...%...........|
[REMOVED]
0944 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0960 77 77 77 2e 64 69 67 74 69 74 61 6c 2d 65 76 69 |www.digtital-evi|
0976 64 65 6e 63 65 2e 6f 72 67 00 00 00 01 00 00 00 |dence.org.......|
0992 01 00 06 00 02 00 04 00 4a 00 00 00 02 00 06 00 |........J.......|
1008 f4 01 00 00 04 00 04 00 10 00 06 00 20 00 04 00 |............ ...|
[REMOVED]
```

رمز تذكري	إزاحة	نوع	
a_version	0x00 (00)	__le32	رقم النسخة !
a_entries[0]	0x04 (04)	posix_acl_xattr_entry	المدخلات !

رمز تذكري	إزاحة	نوع	بنية بيانات المدخلة posix_acl_xattr_entry				
e_tag	0x00 (00)	__le16	حقل النوع (وسم) (tag) في مدخلة ACL يحدد نوع الإذن الذي من أجله كانت المدخلة.:				Type (tag) :
			0x00	- -	- -	ACL_UNDEFINED_TAG	- -
			0x01	Owner	user::rwx	ACL_USER_OBJ	المستخدم المحدد في inode (حقوق النفاذ للمالك الملف)
			0x04	Owning Group	group::rwx	ACL_GROUP_OBJ	المجموعة المالكة (المستخدمون) المحددة في inode (حقوق النفاذ للمجموعة الملف)
			0x20	Other	other::rwx	ACL_OTHER	جميع المستخدمون الآخرون other users (حقوق النفاذ للعمليات التي لا تطابق آية مدخلة أخرى في ACL)
			0x10	Mask	mask::rwx	ACL_MASK	قناع الحقوق النافذة ! (الفعالة) Effective rights mask (حقوق النفاذ الأقصى الممكن منحوها حسب النوع)
			0x02	Named User	user:name:rwx	ACL_USER	المستخدم المسمى المحدد في attribute (حقوق النفاذ للمستخدمين حددهم مصنف ! المدخلة)
			0x08	Named Group	group:name:rwx	ACL_GROUP	المجموعة المسماة (المستخدمون) المحدد في attribute (حقوق النفاذ للمجموعات حددهم مصنف ! المدخلة)
e_perm	0x02 (02)	__le16	أذون النفاذ :				Permission flag :
			0x01	Execute			تنفيذ / بحث
			0x02	Write			كتابة
			0x04	Read			قراءة
e_id	0x04 (04)	__le32	هوية المستخدم / المجموعة User / Group ID (ليس مضمن في بعض الأنواع) (not included for some types)				

```
# istat f linux-ext3 ext3-2.dd 70
[REMOVED]
Extended Attributes (Block: 1238)
user.source=www.digtital-evidence.org
POSIX Access Control List Entries:
uid: 0: Read, Write
uid: 74: Read
uid: 500: Read, Write
gid: 0: Read
mask: Read, Write
other: Read
[REMOVED]
```

هذه الميزة لحماية نظام الملفات من تعدد المضيفين الذين يحاولون استخدام نظام الملفات في نفس وقت (بالتزامن). أي عند فتح نظام الملفات (مثلا، عند وصل، أوعمل فحص fsck،... إلى آخره) شفرة MMP التي تعمل على العقدة (نسميها العقدة "أ") تتفحص رقم متتالية sequence number.إذا كان الرقم هو EXT4_MMP_SEQ_CLEAN، يستمر فتح نظام الملفات.أما إذا كان الرقم هو EXT4_MMP_SEQ_FSCK، حينذاك (على أمل ذلك) تعمل أداة fsck، ويفشل فورا فتح نظام الملفات. ما عدا ذلك، شفرة فتح نظام الملفات سوف تنتظر مرتين فترة الفحص check interval المحددة في MMP وتفحص مرة أخرى رقم المتتالية. إذا تغير الرقم هذا يعني أن نظام الملفات نشيط على جهاز أخر ويفشل فتح نظام الملفات. إذا شفرة MMP اجتازت بنجاح جميع هذه الفحوص، يولد رقم متتالية جديد في MMP ويكتب إلى كتلة MMP، ويستمر الوصل.

أثناء عمل نظام الملفات، النواة تنصب مؤقت timer لإعادة فحص كتلة MMP في فترة الفحص المحددة. ولإعادة الفحص، يعاد قراءة رقم متتالية MMP؛ إذا كان غير متطابق مع رقم متتالية MMP في الذاكرة، حينذاك، تكون عقدة أخرى (العقدة "ب") قد وصلت نظام الملفات، والعقدة "أ" تعيد وصل نظام الملفات في وضعية القراءة فقط. إذا تطابقت أرقام المتتالية، رقم المتتالية يزيد في الذاكرة وعلى القرص، وإعادة الفحص يكتمل.

اسم ملف الجهاز device filename واسم المضيف hostname تكتب في كتلة MMP كلما نجحت عملية فتح للنظام الملفات. شفرة MMP لا تستخدم هذه القيم؛ لأنها موجودة فقط لأغراض معلوماتية !؟.

تدقيق المجموع سيكون بحساب: معرف FS UUID، و بنية MMP.

بنية struct mmp_struct

```
# blkcat /dev/sda5 8940 | hexdump -C
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 [illegible]                                          |PMM.PMM./..\....|
0010 66 69 6e 65 6c 61 6e 64 00 00 00 00 00 00 00 00 00  |fineland........|
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
0050 73 64 61 35 00 73 64 61 35 00 00 00 00 00 00 00 00  |sda5.sda5.......|
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
0070 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
[REMOVED]
03f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
```

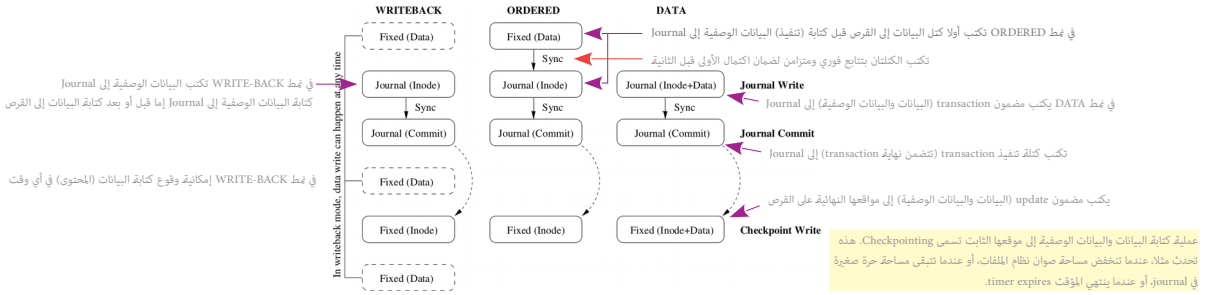
رمز تذكري	إزاحة	حجم / نوع		[REMOVED]		
mmp_magic	0x00 (00)	__le32	4	0x004D4D50U	EXT4_MMP_MAGIC	رقم سحري (التوقيع) من أجل MMP بترميز أسكي "MMP "
mmp_seq	0x04 (04)	__le32	4	رقم المتتالية، الذي يتم تحديثه بشكل دوري (عادة كل 5 ثواني) Sequence no. updated periodically		
				0xFF4D4D50U	EXT4_MMP_SEQ_CLEAN	قيمة نظام الملفات المفصول على نحو نظيف
				0xE24D4D50U	EXT4_MMP_SEQ_FSCK	قيمة عند عمل فحص fsck
				0xE24D4D4FU	EXT4_MMP_SEQ_MAX	أقصى قيمة صالحة في mmp_seq
mmp_time	0x08 (08)	__le64	8	لأغراض معلوماتية فقط، ولا تأثر على صحة الخوارزمية !؟		زمن أخر تحديث للكتلة MMP
mmp_nodename	0x10 (16)	char[64]	64			اسم مضيف العقدة التي فتحت نظام الملفات
mmp_bdevname	0x50 (80)	char[32]	32			اسم جهاز الكتل الخاص بنظام الملفات
mmp_check_interval	0x70 (112)	__le16	2	من أجل التأكد من تحديث كتلة MMP على جهاز الكتل		فترة إعادة تفحص MMP، بعدد الثواني.
mmp_pad1	0x72 (114)	__le16	2			
mmp_pad2	0x74 (116)	__le32[226]	904			
mmp_checksum	0x3FC (1020)	__le32	4	crc32c(uuid+mmp_block)		تدقيق مجموع كتلة MMP

هذه البنية ستكتب إلى رقم الكتلة المحفوظ في s_mmp_block في superblock. البرامج التي تتفحص MMP يجب عليها أن تفترض وجود SEQ_FSCK (أو أية شفرة أعلى SEQ_MAX) وحينذاك يعتبر استخدام نظام الملفات غير آمن، بغض النظر عن قدم الختم الزمني.

```
debugfs: dump_mmp
block_number: 8940
update_interval: 5
check_interval: 5
sequence: ff4d4d50
time: 1559617071 -- Tue Jun  4 02:57:51 2019
node_name: fineland
device_name: sda5
magic: 0x4d4d50
```

تنبيه: حاليا، mmp غير مطبق في ext4 وتمكين هذه الميزة قد يمنع إقلاع النظام (GRUB2)

قيد الحوادث أو journal [137][113] أو log سِجلّ الحوادث ميزة (موجودة في عدة أنظمة ملفات مثل JFS و NTFS) أضيفت في عام 2001 إلى نظام ملفات ext3 من أجل حماية البيانات من التلف. نظام journal يمكن أن يملك خمسة أنواع من الكتل، الأربعة الأولى منها تنفيذية وتعرف بأسماها المميزة ـ Descriptor Block, Commit Block, Revocation Block و Journal Superblock (v1 أو v2) النوع الخامس يسمى عادة Data Block ويخزن البيانات الوصفية metadata فقط أو يخزن البيانات الوصفية و البيانات data التي تم تسجيلها في journal وفقا للنمط عملية قيد الحوادث، كل نوع كتلة تنفيذية يحتفظ بمعلومات ترتبط بنوعه، لكن جميع الكتل التنفيذية الأربعة تتقاسم نفس الصيغة في 12 بايت الأولى. هذه البنية المشتركة تسمى تروسة الكتلة Block Header. النوع الخامس يحتفظ ببيانات وصفية ترتبط بمؤشرات الفهرسة Inodes التي تم تعديلها في نظام الملفات، إن كان journal يستخدم نمط ordered / write back أو يحتفظ بالبيانات الوصفية زائد كتل البيانات (محتوى الملف) إن كان journal يستخدم نمط Data (journaled) الهياكل الداخلية للأنواع المختلفة من الكتل ملخصة في الجداول التالية. أما الشكل التالي فيعرض اختلف تدفق البيانات data flow في كل نمط من أنماط نظام journal.



في هذا الرسم تظهر الأنماط الثلاثة المختلفة من Journal في (ext3/4): write-back و ordered و data (أو Journal). المربعات في الرسم تمثل عمليات التحديث updates (الكتابات) في نظام الملفات، مثلا، "Journal (Inode)" تعني كتابة أحد مؤشرات الفهرسة inode * إلى journal ؛ في هذا الرسم وجهات الكتابة الأخرى تسمى بالثابتة "Fixed" وتعني كتابة (a write) إلى هياكل موقعها ثابت (fixed in-place) في نظام الملفات. السهم مع "Sync" يعني كتابة الكتلتان بتتابع فوري ومتزامن لضمان اكتمال الأولى قبل الثانية. السهم المنحني يشير إلى الترتيب (ordering) لكن بدون تتابع فوري (immediate succession) ـ لذلك ستقع الكتابة الثانية في وقت لاحق. أخيرا، في النمط write-back، المربع بالخط المتقطع حول كتلة "Fixed (Data)" يعني إمكانية وقوعها في المتتالية في أي وقت. في هذا المثال، نعتبر كتابة كتلة البيانات مع inode (بياناتها الوصفية) كعمليات updates تحتاج إلى تنفيذ/نقل (propagated) إلى نظام الملفات.

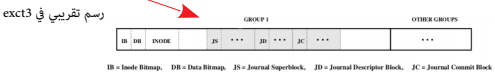
ext4 يخصص منطقة متواصلة صغيرة على القرص (مبدئيا 128 ميغابايت) داخل نظام الملفات كموضع سريع للكتابة البيانات المهمة على القرص. بمجرد كتابة كامل إجراء [84][136] البيانات المهمة (أي مجموعة updates) إلى القرص وتخليصها [123] من ذاكرة التخزين المؤقت للكتابة القرص [124] disk write cache، يتم أيضا كتابة سجل [22] بالبيانات المنفذة إلى journal. في وقت لاحق، شفرة journal ستكتب الإجراءات إلى موقعهم النهائي على القرص (قد ينطوي هذا على الكثير من السعي "أي تحريك رأس القراءة/الكتابة" أو الكثير من العمليات الصغرى؛ مسح-كتابة-قراءة read-write-erases) قبل مسح سجل التنفيذ commit record. في حالة انهيار النظام أثناء عملية الكتابة البطيئة الثانية، يمكن إعادة تشغيل journal بالكامل حتى آخر سجل تنفيذ لضمان ذرية [46]كل ما يكتب من خلال journal إلى القرص. هذا يضمن عدم توقف النظام في منتصف الطريق عند تحديث metadata

لأسباب تتعلق بالأداء، ext4 يكتب فقط البيانات الوصفية للنظام الملفات من خلال journal. هذا يعني أن ثبات كتل بيانات الملف ليس مضمون في حال انهيار النظام. لكن إذا كان مستوى الضمان الاعتيادي data=ordered غير مرضي، هناك خيار للوصل يمكن من خلاله التحكم في سلوك قيد الحوادث في زمن إعادة وصل نظام الملفات (راجع دليل استخدام mount).

إن كان النمط هو data=journal، جميع البيانات والبيانات الوصفية (Inode+Data) تكتب إلى القرص من خلال journal. هذا الخيار أبطأ لكنه الأكثر أمانا، وسيعطل دعم delayed allocation و O_DIRECT. (تنبيه: في هذا النمط لن تستطيع وصل قسم ext4 في حالة تمكين ميزة delayed allocation).

إن كان النمط هو data=writeback، لا يتم تخليص [123] كتل البيانات الملوثة (معدلة!) [26] dirty data blocks إلى القرص قبل كتابة metadata إلى القرص من خلال journal. هذا سيضمن تكامل نظام الملفات داخليا، لكن البيانات الفيزيائية data قد تتلف. هذا يحدث إذا وقع الانهيار بعد إنشاء metadata في journal، وقبل كتابة البيانات الفيزيائية. (أي هنا ترتيب البيانات غير محفوظ).

مؤشر فهرسة journal inode عادة رقم 8 وبدون مدخلة دليل (أي مخفي hard-coded) وأول 68 بايت من journal inode مكررة في الكتلة العليا [112] ext4 superblock. قيد الحوادث نفسه عبارة عن ملف اعتيادي (سجل دوري**ـ مؤقت المحتوى ومخفي) داخل نظام الملفات. حجم ملف journal سيكون بناء على حجم نظام الملفات، لكنه عادة يستهلك مساحة مجموعة كتل كاملة، مبدئيا ext3 كان ينشئ journal في بداية القسم، وفي ext4 برنامج mke2fs (الذي يستخدم extents) سيحاول وضع الملف في منتصف نظام ملفات ext4 (القسم) للتقليل من زمن السعي seek time.



حقول jbd2 تكتب إلى القرص بترتيب نهوي كبير، (عكس المعمول به في ext4. الذي يعمل بترتيب نهوي صغير) لكن عند البحث عن إحدى كتل المحتوى أو البيانات الوصفية الترقيم يعتمد على النظام الذي أنشأه.

نظام ملفات ocfs2 ونظام ملفات ext4 كلاهما يستخدم jbd2 [53] طبقة المضمن في ext4 حجم journal الأقصى هو 2^{32} كتلة. ويبدو أن Jbd2 نفسه لا يكترث لذلك.

* في الإعدادات الاعتيادية، فقط التغييرات في metadata تسجل في journal. هذا يعني عند تعديل مثلا ملف، الكتل التي تسجل في journal هي group descriptor table و inode لمدخلة الدليل ومدخلة الدليل directory entry للدليل الذي يتضمن الملف و inode الخاص بالملف و inode bitmap و block bitmap لمجموعة الكتل block group التي ينتمي إليها ذلك الملف. وطبعا تسجل أيضا كتل مضمون ذلك الملف في حالة تمكين نمط data=journal. التغيرات التي يمكن أن تحدث في Journal في هذا المثال: [141]

نظرا لأن Journal يعمل على مستوى الكتلة block level، تحديث 1 بت في inode يعني نسخ كامل كتلة نظام الملفات inode إلى journal بمعنى آخر inodes في نفس الكتلة تنسخ أيضا إلى journal وهذا ما سماه Dan Farmer و Witse Venema بتأثير المتفرج! bystander effect

** كسجل دائري أو circular list أو صوان حلقي ring buffer يعيد الكتابة فوق البيانات القديمة فقط، هذا صحيح طالما النظام في حالة وصل. لكن عند إعادة وصل النظام أو عند امتلئ journal هذا الأخير سيكتب من بداية أول كتلة، هذه الآلية ستدمر أية بيانات سابقة. إن كان هناك وصل جديد، نظام journal لا يكتب أبعد من الموقع الأخير، هذا يعني الكتابة فوق البيانات الموجودة حتى وإن كانت البيانات الأقدم ما زالت في وسط أو في نهاية journal. خصوصا عند وصل أو إعادة تشغيل النظام على فترات قصيرة، الذي يعني الكتابة فوق البيانات التي كانت من الوصل الأخير. في نهاية journal ستكون هناك بيانات قديمة جدا وغير متاحة مع ثغرات زمنية.

تخطيط قيد الحوادث

Layout

عموما، قيد الحوادث [113] سيكون بالصيغة التالية:

[المزيد من الإجراءات ...] [more transactions...]	كتلة_توصيف (كتل_بيانات أو كتلة_إلغاء) [المزيد من البيانات أو الإلغاء] كتلة_تنفيذ descriptor_block (data_blocks or revocation_block) [more data or revocations] commit_block One transaction إجراء واحد	كتلة عليا Journal Superblock

لحظ أن الإجراء يبدأ إما بواصف وبعض البيانات أو لائحة إلغاءات للكتل block revocation list والإجراء المكتمل دائما ينتهي بتنفيذ commit. إذا لم يكن هناك سجل تنفيذ commit record (أو تدقيق المجامع لا يتطابق) سيتم تجاهل الإجراء أثناء إعادة تشغيل journal

قيد الحوادث الخارجي

External Journal

يمكن أيضا إنشاء نظام ملفات ext4 مع جهاز قيد حوادث خارجي [114] (هذا عكس قيد الحوادث الداخلي internal journal، الذي يستخدم inode المحجوز) في هذه الحالة، حقل s_journal_inum، في جهاز نظام الملفات يجب أن يكون صفر ويجب تعيين s_journal_uuid. وستكون على جهاز قيد الحوادث، كتلة ext4 super block في المكان الاعتيادي مع معرف UUID. والكتلة العليا في قيد الحوادث ستشغل كامل الكتلة التالية بعد الكتلة الاعتيادية superblock.

[المزيد من الإجراءات ...] [more transactions...]	كتلة_توصيف (كتل_بيانات أو كتلة_إلغاء) [المزيد من البيانات أو الإلغاء] كتلة_تنفيذ descriptor_block (data_blocks or revocation_block) [more data or revocations] commit_block One transaction إجراء واحد	الكتلة العليا في قيد الحوادث Journal Superblock	الكتلة العليا ext4 Superblock	1024 بايت حشو [98] 1024 bytes of padding

ترويسة الكتلة

Block Header

كل كتلة في قيد الحوادث تبدأ بترويسة 12-بايت struct journal_header_s (هذه الترويسة معيارية في جميع كتل التوصيف التنفيذية descriptor blocks):

ترويسة قيد الحوادث struct journal_header_s (بحجم 12 بايت)

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 c0 3b 39 98 00 00 00 01 00 0c d5 cd 00 00 00 00 |.;9.............|
```

[REMOVED]

رقم سحري (التوقيع) من أجل jbd2	0xC03B3998	jbd2 magic number	4	حجم / نوع	إزاحة	رمز تذكري
h_magic	0x00 (00)	__be32	4			
نوع كتلة التوصيف / وصف مضمون هذه الكتلة (journal block type) سيكون إحدى هذه :		Descriptor block types :				
توصيف. هذه الكتلة تسبق سلسلة من كتل البيانات التي كتبت من خلال قيد الحوادث أثناء الإجراء.	Descriptor Block	JBD2_DESCRIPTOR_BLOCK	1			
سجل تنفيذ الكتلة. هذه الكتلة تدل على اكتمال الإجراء.	Commit Block	JBD2_COMMIT_BLOCK	2			
الكتلة العليا في قيد الحوادث، النسخة 1	Journal superblock, v1	JBD2_SUPERBLOCK_V1	3			
الكتلة العليا في قيد الحوادث، النسخة 2	Journal superblock, v2	JBD2_SUPERBLOCK_V2	4			
تسجيلات إلغاء الكتل. هذه تسرع الاسترداد بتمكين Journal من تخطي كتابة الكتل التي يعاد كتبتها لاحقا	Revocation Block	JBD2_REVOKE_BLOCK	5			
هوية الإجراء المصاحب لهذه الكتلة. transaction ID						

v1 (أول 36 بايت) تسجل جميع data updates، و v2 تسجل فقط metadata updates

الكتلة العليا

Super Block

الكتلة العليا في قيد الحوادث journal أبسط بكثير مقارنة بالكتلة العليا للنظام الملفات [112] ext4 super block. البيانات المهمة في هذه الكتلة ستكون حجم journal، ومكان بداية سجل الإجراءات transactions (جميع الحقول بترتيب بايت نهوي كبير)

journal superblock مسجلة في struct journal_superblock_s (بحجم 1024 بايت)

```
          dd if=journal.dd bs=1024 count=1 | hexdump -C
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 c0 3b 39 98 00 00 00 04 00 00 00 00 00 00 10 00 |.;9.............|
0010 00 00 80 00 00 00 00 01 00 0c d5 f5 00 00 00 00 |................|
0020 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 |................|
0030 2d 45 37 64 6b 5a 48 f7 b0 be fc 78 c0 40 bf 91 |-E7dkZH....x.@..|
0040 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 10 |................|
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[REMOVED]
00f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
```

[REMOVED]

معلومات ثابتة تصف قيد الحوادث journal						
ترويسة عامة، من أجل التعريف بالكتلة العليا superblock	(h_magic + h_blocktype + h_sequence)			journal_header_t (بايت 12)	0x00 (00)	s_header
حجم كتلة جهاز قيد الحوادث	journal device blocksize	4	__be32		0x0C (12)	s_blocksize
عدد الكتل الإجمالي في ملف قيد الحوادث	total blocks in journal file	4	__be32		0x10 (16)	s_maxlen
أول كتلة من معلومات السجل	first block of log information (start of the journal entries)	4	__be32		0x14 (20)	s_first
معلومات ديناميكية تصف الوضعية الحالية للسجل						
أول هوية تنفيذ متوقع في السجل	first commit ID expected in log (first sequence number/first transaction)	4	__be32		0x18 (24)	s_sequence
رقم كتلة بداية السجل (0 في هذا الحقل لا يعني أن قيد الحوادث نظيف!)	blocknr of start of log	4	__be32		0x1C (28)	s_start
قيمة الخطأ، كما تم تعيينها بواسطة jbd2_journal_abort ()	Error value, as set by journal_abort()	4	__be32		0x20 (32)	s_errno
بقية الحقول صالحة فقط في النسخة 2 من superblock.						
مجموعة أعلام الميزات المتوافقة [27]	Compatible feature set :	4	__be32		0x24 (36)	s_feature_compat;
قيد الحوادث يحفظ تدقيق المجاميع على كتل البيانات	JBD2_FEATURE_COMPAT_CHECKSUM	0x01				
مجموعة أعلام الميزات الغير متوافقة	Incompatible feature set :	4	__be32		0x28 (40)	s_feature_incompat
قيد الحوادث يملك تسجيلات إلغاء الكتلة block revocation records	JBD2_FEATURE_INCOMPAT_REVOKE	0x01				
قيد الحوادث يستطيع التعامل مع أرقام / أعداد كتلة 64-بت	JBD2_FEATURE_INCOMPAT_64BIT	0x02				
تنفيذ قيد الحوادث لا تزامني	JBD2_FEATURE_INCOMPAT_ASYNC_COMMIT	0x04				
قيد الحوادث هذا يستخدم النسخة v2 من صيغة تدقيق المجموع، على القرص [93]	JBD2_FEATURE_INCOMPAT_CSUM_V2	0x08				
قيد الحوادث هذا يستخدم النسخة v3 من صيغة تدقيق المجموع على القرص. [94]	JBD2_FEATURE_INCOMPAT_CSUM_V3	0x10				
مجموعة أعلام الميزات المتوافقة- في وضعية للقراءة فقط [27] حاليا هذه لا توجد.	Read-only compatible feature set :	4	__be32		0x2C (44)	s_feature_ro_compat
--	--	--				
معرف UUID 128-بت من أجل قيد الحوادث. هذا يقارن بالنسخة في ext4 super block في زمن وصل نظام الملفات.		16	__u8		0x30 (48)	s_uuid[16]
عدد أنظمة الملفات التي تتشارك قيد الحوادث هذا		4	__be32		0x40 (64)	s_nr_users
موقع نسخة الكتلة العليا الديناميكية dynamic super block (غير مستخدم؟)		4	__be32		0x44 (68)	s_dynsuper
حد عدد كتل قيد الحوادث لكل إجراء. (غير مستخدم؟) Limit of journal blocks per transaction		4	__be32		0x48 (72)	s_max_transaction
حد عدد كتل البيانات لكل إجراء. (غير مستخدم؟) Limit of data blocks per transaction		4	__be32		0x4C (76)	s_max_trans_data
نوع خوارزمية تدقيق المجموع المستخدمة من أجل قيد الحوادث، الأرجح ستكون 1 أو 4	Checksum algorithm :	1	__u8		0x50 (80)	s_checksum_type
	JBD2_CRC32_CHKSUM	crc32		1		
	JBD2_MD5_CHKSUM	md5		2		

				3	JBD2_SHA1_CHKSUM	sha1
				4	JBD2_CRC32C_CHKSUM	crc32c
s_padding2	0x51 (81)	__u8[3]	3			
s_padding[42]	0x54 (84)	__u32	168			
s_checksum	0xFC (252)	__be32	4	crc32c(superblock)		تدقيق مجموع كامل superblock، مع تعيين هذا الحقل إلى الصفر.
s_users[16*48]	0x100 (256)	__u8	768			ids جميع أنظمة الملفات التي تتشارك هذا السجل (e2fsprogs/Linux لا تسمح بقيود الحوادث الخارجية المشتركة، لكن Lustre أو ocfs2؟ التي تستخدم jbd2.قد تسمح).

كتلة توصيف

Descriptor Block

كتلة التوصيف تتضمن مصفوفة من[129] journal block tags تصف المواقع النهائية للكتل البيانات التي تتبع في قيد الحوادث. **كتل التوصيف** ستكون بترميز مفتوح open-coded [35] بدل وصفها بالكامل بواسطة بنية بيانات، على أي حال هذه البينة:

كتل التوصيف تستهلك على الأقل 36 بايت، (لكن تحجز كتلة كاملة)

```
# jcat -f ext3 dev/hdb2 1 | hexdump -C
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000  c0 3b 39 98 00 00 00 01  00 00 01 27 00 00 00 04  |.;9........'....|
0016  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0032  00 00 00 00 00 00 00 02  00 00 00 02 00 00 00 0e  |................|
0048  00 00 00 02 00 00 00 05  00 00 00 02 00 00 00 a3  |................|
0064  00 00 00 02 00 00 00 03  00 00 00 0e 00 00 00 00  |................|
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

[REMOVED]

رمز تذكري	إزاحة	حجم / نوع			
(open coded)	0x00 (00)	journal_header_t	12	(h_magic + h_blocktype + h_sequence)	ترويسة كتلة عامة
open coded array[]	0x0C (12)	struct journal_block_tag_s	--		لواحق tags (هياكل صغرى، تسمى أيضا: مدخلات entries) تكفي لملء الكتلة أو تكفي لوصف جميع كتل البيانات التي تتبع كتلة التوصيف هذه.

لاحقة الكتلة block tag تستخدم في وصف صوان buffer واحد في journal. لواحق كتلة قيد الحوادث journal block tags سوف تملك إحدى الصيغ التالية، بناء على ميزة قيد الحوادث وأعلام لاحقة الكتلة block tag التي تم تعيينها.

رمز تذكري	إزاحة	حجم / نوع				
				عند تعيين JBD2_FEATURE_INCOMPAT_CSUM_V3، لاحقة كتلة قيد الحوادث ستكون struct journal_block_tag3_s (بحجم 16 أو 32 بايت)		
t_blocknr	0x00 (00)	__be32	4	(File system block ↔ Journal block)	رقم الكتلة على القرص / الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص. (32-بت المنخفضة)	
t_flags	0x04 (04)	__be32	4	the journal tag flags word / Entry flags :		الأعلام التي تأتي مع التوصيف descriptor، ستكون أيا من هذه:
				0x01	JBD2_FLAG_ESCAPE	تخطي الكتلة على القرص. عند تطابق أول 4 بايت من كتلة البيانات مع الرقم السحري (التوقيع) في ترويسة jbd2
				0x02	JBD2_FLAG_SAME_UUID	هذه الكتلة تملك نفس معرف UUID،مثل السابقة، ولذلك، تم إسقاط حقل UUID
				0x04	JBD2_FLAG_DELETED	كتلة البيانات حذفها هذا الإجراء (غير مستخدم؟)
				0x08	JBD2_FLAG_LAST_TAG	هذه أخر لاحقة tag في كتلة التوصيف هذه
t_blocknr_high	0x08 (08)	__be32	4			الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص (32-بت العليا) هذه صفر في حالة تعطيل JBD2_FEATURE_INCOMPAT_64BIT
t_checksum	0x0C (12)	__be32	4	crc32c(uuid+seq+block)		تدقيق مجموع كل من معرف قيد الحوادث UUID، ورقم المتتالية، وكتلة البيانات.
				هذا الحقل يبدو بترميز مفتوح [35] open coded. ويأتي دائما في نهاية اللاحقة Tag. بعد t_checksum. هذا الحقل لن يكون موجود في حالة تعيين علم "SAME_UUID"		
uuid[16]	0x08 / 0x0C	char	16			UUID الذي يأتي مع هذه اللاحقة Tag. هذا الحقل يبدو منسوخ من حقل j_uuid في struct journal_s، لكن فقط tune2fs يمكنها التعامل مع ذلك الحقل

رمز تذكري	إزاحة	حجم / نوع				
				عند تعطيل JBD2_FEATURE_INCOMPAT_CSUM_V3 لاحقة كتلة قيد الحوادث ستكون struct journal_block_tag_s (بحجم 8، 12، 24 أو 28 بايت)		
t_blocknr	0x00 (00)	__be32	4			رقم الكتلة على القرص / الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص.(32-بت المنخفضة)
t_checksum	0x04 (04)	__be16	2	truncated crc32c(uuid+seq+block)		تدقيق مجموع: معرف journal UUID، ورقم المتتالية وكتلة البيانات لاحظ هنا: تخزن فقط 16 بايت المنخفضة
t_flags	0x06 (06)	__be16	2	the journal tag flags word :		الأعلام التي تأتي مع التوصيف descriptor. أيا من هذه:
				0x01	JBD2_FLAG_ESCAPE	تخطي الكتلة على القرص، فقط في حالة تطابق أول 4 بايت من كتلة البيانات مع الرقم السحري (التوقيع) في jbd2
				0x02	JBD2_FLAG_SAME_UUID	هذه الكتلة، تملك نفس معرف UUID، مثل السابقة ولذلك تم إسقاط حقل UUID
				0x04	JBD2_FLAG_DELETED	كتلة البيانات حذفها إجراء. (غير مستخدم؟)
				0x08	JBD2_FLAG_LAST_TAG	هذه أخر لاحقة في كتلة التوصيف هذه
				الحقل التالي سيكون موجود فقط في حالة كانت الكتلة العليا super block تشير إلى دعم أرقام / أعداد كتلة 64-بت		
t_blocknr_high	0x08 (08)	__be32	4			الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص. (32-بت العليا)
				هذا الحقل يبدو مفتوح الترميز open coded. ويأتي دائما في نهاية اللاحقة Tag بعد t_flags أو t_blocknr_high. هذا الحقل لن يكون موجود في حالة تعيين علم نفس المعرف "SAME_UUID"		
uuid[16]	0x08 / 0x0C	char	16			UUID الذي يأتي مع هذه اللاحقة Tag . هذا الحقل يبدو منسوخ من حقل j_uuid في struct journal_s، لكن فقط tune2fs تستطيع التعامل مع ذلك الحقل

رمز تذكري	إزاحة	حجم / نوع			
				عند تعيين CSUM_V2 أو CSUM_V3 نهاية الكتلة تكون struct jbd2_journal_block_tail (هذا سيكون ذيل كتلة revoke أو descriptor، للحساب تدقيق المجموع)	
t_checksum	0x00 (00)	__be32	4	crc32c(uuid+descr_block)	تدقيق مجموع: معرف قيد الحوادث journal UUID + كتلة التوصيف، مع تعيين هذا الحقل إلى الصفر

إذا انطوى الإجراء transaction على كتل blocks كثيرة بحيث لا يمكن وصفها في كتلة توصيف واحدة، يتم إنشاء كتلة توصيف Descriptor Block أخرى لاستيعاب بقية الكتل. وكتلة التنفيذ Commit Block تظهر فقط في نهاية الإجراء.

كتلة البيانات

Data Block

عموما، كتل البيانات التي تكتب إلى القرص من خلال قيد الحوادث تكتب حرفيا داخل ملف قيد الحوادث بعد كتلة التوصيف. لكن، في حالة تطابق 4 بايت الأولى من الكتلة مع الرقم السحري (التوقيع) في jbd2 حين ذلك، تستبدل هذه 4 بايت بأصفار ويتم تعيين علم "escaped" في لاحقة كتلة التوصيف descriptor block tag.

كتلة الإبطال (الإلغاء / النقض...)

Revocation Block

كتلة الإبطال تستخدم لمنع تكرار replay كتلة في إجراء سابق. هذا revoke descriptor يصف سلسلة من الكتل ستلغى من log (في الإجراء الحالي) بمعنى آخر، يستخدم لتعليم الكتل التي كتبت إلى قيد الحوادث سابقا ولم تعد في Journal. عادة، هذا يحدث إذا كانت كتلة البيانات الوصفية حرة (metadata block) ثم خصصت مرة ثانية ككتلة بيانات ملف (data block)؛ في هذه الحالة، تكرار journal بعد كتابة كتلة الملف إلى القرص سوف يتسبب في تلف البيانات.

تنبيه: استخدام هذه الآلية لا يعني أن "كتلة قيد الحوادث هذه قد ألغتها كتلة قيد الحوادث الأخرى" أية كتلة يتم إضافتها إلى إجراء ستتسبب في إزالة جميع تسجيلات الإلغاء revocation records الموجودة لتلك الكتلة

كتل الإبطال Revocation blocks موصوفة في struct jbd2_journal_revoke_header_s، (بطول 16 بايت على الأقل، لكن تحتل كتلة كاملة)

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 c0 3b 39 98 00 00 00 05 00 0c d5 cd 00 00 00 28 |.;9............(|
0010 00 20 32 80 00 20 24 2c 00 20 25 f3 00 20 35 27 |. 2.. $,. %.. 5'|
0020 00 20 32 83 00 20 2e 1d 00 00 00 00 00 00 00 00 |. 2.. ..........|
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[REMOVED]
0ff0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[REMOVED]
```

رمز تذكري	إزاحة	نوع / حجم		
r_header	0x00 (00)	journal_header_t	12	(h_magic + h_blocktype + h_sequence) — ترويسة كتلة عامة
r_count	0x0C (12)	__be32	4	عدد بايتات المستخدمة في هذه الكتلة
Blocks[0]	0x10 (16)	__be32 / __be64	--	لائحة من عناوين كتل نظام الملفات للإلغاء Blocks to revoke

بعد حقل r_count ستكون مصفوفة خطية من أرقام الكتل الملغى فعليا من قبل هذا الإجراء، حجم كل رقم كتلة 8 بايت في حالة أعلنت superblock عن دعم رقم كتلة 64-بت، أو خلاف ذلك ستكون 4 بايت.

رمز تذكري	إزاحة	نوع / حجم		
				في حال تعيين JBD2_FEATURE_INCOMPAT_CSUM_V2 أو JBD2_FEATURE_INCOMPAT_CSUM_V3، نهاية كتلة الإلغاء ستكون struct jbd2_journal_revoke_tail
r_checksum	0x00 (00)	__be32	4	crc32c(uuid+REVOKE_BLOCK) — تدقيق مجموع كل من معرف قيد الحوادث UUID journal+كتلة الإبطال (الإلغاء)

كتلة التنفيذ (أو الإيداع)

Commit Block

كتلة التنفيذ commit block ستكون حارس يشير إلى اكتمال كتابة الإجراء إلى قيد الحوادث Journal. عندما تصل كتلة التنفيذ هذه إلى Journal، البيانات المخزنة مع هذا الإجراء transaction يمكن كتابتها إلى مواقعها النهائية على القرص.

كتلة التنفيذ commit block موصوفة في struct commit_header (بطول 32 بايت لكن تستخدم كتلة كاملة)

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 c0 3b 39 98 00 00 00 02 00 0c cc 5c 00 00 00 30 |.;9........\...0|
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0030 00 00 00 00 5b 52 9a 46 08 e5 dc f2 00 00 00 00 |....[R.F........|
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[REMOVED]
```

رمز تذكري	إزاحة	نوع / حجم			
(open coded)	0x00 (00)	journal_header_s	12	(h_magic + h_blocktype + h_sequence)	ترويسة كتلة عامة
h_chksum_type	0x0C (12)	unsigned char	1	Checksum types : نوع تدقيق المجموع المستخدم للتأكد من تكامل كتل البيانات في الإجراء. وستكون إحدى هذه: [87]	
				1 JBD2_CRC32_CHKSUM crc32	
				2 JBD2_MD5_CHKSUM md5	
				3 JBD2_SHA1_CHKSUM sha1	
				4 JBD2_CRC32C_CHKSUM crc32c	
h_chksum_size	0x0D (13)	unsigned char	1	JBD2_CRC32_CHKSUM_SIZE = 4	عدد بايتات المستخدمة من قبل تدقيق المجموع. الأرجح ستكون 4.
h_padding[2]	0x0E (14)	unsigned char	2		
h_chksum[JBD2_CHECKSUM_BYTES]	0x10 (16)	__be32	32		مساحة من 32 بايت من أجل تخزين تدقيق المجامع [89]
h_commit_sec	0x30 (48)	__be64	8	أختام زمنية	زمن تنفيذ الإجراء، بعدد الثواني (توقيت يونكس)
h_commit_nsec	0x38 (56)	__be32	4		مكون نانو ثانية من الختم الزمني الذي في الأعلى. Nanoseconds component

ملاحظات:

1. ^ أ، ب، ت، ث، في أنظمة م.س دوس، البنية التي تتعقب الملف، تسمى سجل الملف أو كتلة التحكم بالملفات FCB. في لينكس وشبيه يونكس، هذه البنية تدعى inode (مركبة من index node وتعني مؤشر الفهرسة! أو عقدة الملف!) تتعقب الملف أو ملف الدليل (أي الدليل)، وتتضمن جميع البيانات الوصفية للملف (باستثناء اسم ومضمون الملف) بنية inode يمكن أن تتضمن أيضا بيانات مباشرة في حالة تمكين ميزة INLINE_DATA، وكان مضمون الملف أقل من 60 بايت، أو تتضمن وصلات Symbolic Links، إذا كانت سلسلة الهدف (أي الاسم) أقل من 60 بايت، ما عدا ذلك، حقل 60 بايت سيكون مخصص للتخزين Extent Tree في ext4 (أو عنونة الكتل المباشرة والغير مباشرة في ext2/3 لأن 60 ÷ 4 = 15). عدد inodes الإجمالي مع المساحة المحجوزة لها يتحدد عند إنشاء نظام الملفات (حوالي 1% تكرس من أجل inodes أو inode واحد لكل 16KB من كتل البيانات). حد inodes لا يتغير ديناميكياً، وكل كائن (أي ملف، دليل، إلى آخره) في نظام الملفات يجب أن يملك هذه البنية inode التي عددها ثابت ويشير إلى عدد الملفات الأقصى في كل نظام ملفات. مثال:

```
# tune2fs -l /dev/sda1 | grep  inode
Inodes per group: 8144
Inode blocks per group: 509
Inode size: 256
```

حجم جدول مؤشرات الفهرسة في كل مجموعة كتل 509، هذا الرقم مرتبط برقمين هما، حجم مؤشر الفهرسة 256 بايت (المستخدم في ext4)، وعدد مؤشرات الفهرسة لكل مجموعة، 8144 (في هذا المثال)،

إذن عدد كتل inodes في كل مجموعة سيكون بحساب : 256 بايت × 8144 ÷ 4096 بايت = 509

لكن هناك احتمالية (قد تبدوا غريبة) أن تنفد inodes على القرص، إذا حدث ذلك، المستخدم لن يستطيع إنشاء ملفات جديدة، حتى وإن كانت هناك مساحة حرة متوفرة على القرص. هذه الحالة يمكن أن تحدث في خوادم البريد التي تتضمن الكثير من الملفات الصغرى

```
# df -i
Filesystem     Inodes IUsed  IFree IUse%  Mounted on
/dev/sda1     1310720 1310720     0  100%  /
```

على أية حال، النقص في inodes يمكن أن يحدث في الحالات التالية:

- عند إنشاء عدد كبير من الأدلة، وصلات الرمزية، الملفات الصغرى...
- عند إنشاء نظام الملفات باستخدام حجم كتلة أصغر. إذا كان نظام الملفات يستعمل في تخزين الكثير من الملفات الصغرى، من المحتمل أن يكون حجم كتلة 1024 أو 4048 بايت. هذا سيسمح باستغلال مساحة القرص بشكل فعال، لكنه أيضا يزيد في إمكانية استهلاك inodes.

  سؤال: لا أريد أو لا أستطيع حذف الملفات الموجودة فهل أستطيع رفع حد inodes بطرق أخرى؟

  الجواب: في أنظمة ملفات ext ، لا تستطيع بباسطة رفع حد inodes على وحدة التخزين الموجودة فعليا. وأنت أمام خيارين:

  - إذا كان القرص LVM، يمكنك زيادة حجم وحدة التخزين
  - أو عمل نسخ احتياطي ثم إنشاء نظام ملفات جديد، مع تحديد حد أعلى للمؤشرات الفهرسة عن طريق الخيار: N- mke2fs (راجع man mke2fs) قبل ذلك ينصح بتنفيذ ومراجع معلومات الخيار n-
- في حالة كان الخادوم داخل حاوية (Docker, LXC, OpenVZ ..إلى آخره) الخوادم داخل حاويات غالبا ما تشترك في نفس نظام الملفات مثل عقدة الجهاز المضيف host node. لغرض الاستقرار والأمان، موارد الحاويات مثل RAM، و CPU ومساحة القرص و inodes ستكون محدودة. في هذه الحالة، عدد inodes المخصص للحاوية يقرره مدير host node. وحدوث مشاكل مع inodes في الحاويات شائع جدا مع أنظمة الملفات من هذا النوع. لكن أنظمة ملفات (مثل Btrfs ، XFS ، JFS) تستطيع تجاوز هذا التقييد باستخدام المديات و/أو تخصيص inodes ديناميكية، هذا ينمي نظام الملفات و/أو يرفع عدد inodes. بالمناسبة هناك أنظمة ملفات لا تستخدم أصلا inodes مثل نظام ملفات ZFS .
2. ^ رغم أن هذا الاحتمال بعيد لكنه ممكن، ماذا سيحدث إذا أعداد الفهارس/ المواقع (مثلا في توصيف المجموعات) وصلت إلى أقصى قيمة لها وهي 32-بت ؟

   إذا كنت تملك مساحة تخزين كبيرة (بعدد كتل أكبر من 2^32) نظام الملفات ext4 سيستخدم نمط 64-بت. والأعداد المهمة تصبح بقيم 64-بت. أيضا يمكنك تمكين الميزة عن طريق tune2fs -o 64bit.

   يمكنك أيضا تحويل نظام الملفات الحالي إلى نظام 64 بت مع تمكين ميزة تدقيق مجموع، لكن أولا، تحتاج إلى فحص وأمثلة القسم باستخدام e2fsck (بدون وصل القسم) ثم عمل بقية الخطوات:

```
# e2fsck -Df /dev/sda1  (فحص إجباري للنظام الملفات مع أمثلة الأدلة)
Convert the filesystem to 64bit:
# resize2fs -b /dev/sda1  (تمكين ميزة 64 بت مع إعادة تحجيم توصيف المجموعات عند الضرورة، وتحريك البيانات الوصفية الأخرى)
Finally enable checksums support:
# tune2fs -O metadata_csum /dev/sda1  (تخزين تدقيق المجموع لحماية المحتوى في كل كتلة بيانات وصفية)
```

3. ^ أظن أن "EF" في توقيع 0xEF53 تشير إلى "Extended Filesystem" و 53 رقم إصدارة ! (تنبيه: 0xEF53 ليست دائما إشارة صحيحة راجع أمثلة ef53 -l 56 –sigfind -0 في كتاب"File System Forensic Analysis")

   إزاحة التوقيع عند 0x38 (أي 56 بايت بعد 1024 بايت المحجوزة) أو تحديدا الموقع (00000438) في السطر: ((67 × 16) = 0x00000430

```
# dd if=/dev/sda1 bs=16 skip=67 count=1 | hexdump -Cv
# hexdump -C -n 4096 /dev/sda1 | grep "53 ef"
 0430 a9 10 e7 54 01 00 ff ff 53 ef 01 00 01 00 00 00 |...T....S.......|
```

في المثال 1072 = 0x430 ، دليل على وجود 1 كيلوبايت المحجوزة من أجل شفرات الإقلاع مثل شفرة VBR

4. ^ إذا لم يكن تعيين الإصدارة الرئيسية إلى 2 (ديناميكية) القيم من بايتات 84 فصاعدا قد تكون غير صحيحة. "ديناميكية" هنا تعني أن كل inode يمكن أن يكون بحجم متغير، والحجم الفعلي مخزن في superblock في بايتات 88 و 89
5. ^ التهيئة الكسولة أو التهيئة المؤجلة أو التهيئة عند الحاجة Lazy initialization (نوع من التقييم الكسول Lazy Evaluation).

   يتم فيه تأجيل إنشاء الغرض أو حساب قيمة أو عملية ما إلى وقت لاحق بحيث تكون تكلفة هذه العملية كبيرة لذا يتم تأجيلها إلى أن يصبح هناك حاجة لها. هنا LAZY تعني إنشاء نظام الملفات بدون تهيئة جميع المجموعات (لزيادة سرعة إنشاء نظام الملفات). لان النواة فيما بعد ستكمل تهيئة نظام الملفات في الخلفية، عند وصل نظام الملفات أول مرة، و ext4 يدعم هذه الميزة في ألأنويه الحديثة.
6. ^ ميزة تمكن المدير من تغيير UUID الخاص بميزة نظام ملفات metadata_csum أثناء وصل نظام الملفات؛ وبدونها تعريف تدقيق المجموع يتطلب إعادة كتابة جميع كتل البيانات الوصفية
7. ^ قبل هذه الميزة، كانت الأدلة لا تتجاوز 4 جيجابايت ولا يمكنها أن تملك شجرة HTree بمستوى أعمق من 2، في حالة تمكين الميزة، الأدلة يمكنها تجاوز 4 جيجابايت وتملك HTree بعمق أقصى 3
8. ^ بالإضافة إلى كشف تلف السانات، هذا مفيد للتهيئة المؤجلة مع المجموعات uninitialized groups (راجع أعلام فقرة Lazy Block Group Initialization).
9. ^ في المراجعة 0 قيمة 32 بت هذه دائما 0، وفي المراجعة 1، من أجل الملفات الاعتيادية هذه القيمة تتضمن 32 بت العليا من حجم ملف 64 بت.

   ملاحظة: لينكس يعين هذه إلى 0 إذا كان الملف ليس ملف اعتيادي (أي ملف أجهزة كتلة، أدلة، ..إلى آخره) نظريا، يمكن تعيين القيمة للإشارة إلى الكتلة المتضمنة خصائص ممتدة للدليل أو الملف الخاص
10. ^ أ، ب، ت، ث، ج، بنية مدخلة الدليل الموصولة Linked Directory Entry Structure (في المراجعة 0.5 والمراجعات اللاحقة):
    - حقل inode: قيمة 32-بت تشير إلى رقم مؤشر فهرسة الملف inode number. القيمة 0 هنا تشير إلى أن المدخلة غير مستخدمة
    - حقل طول التسجيلة rec_len:.قيمة 16 بت تشير إلى إزاحة لا تحمل إشارة إلى مدخلة الدليل التالية من بداية المدخلة الحالية. قيمة هذا الحقل يجب أن تساوي على الأقل طول التسجيلة الحالية محاذاة مدخلات الدليل يجب أن تكون على حدود 4 بايت ولا يمكن لأية مدخلة دليل الامتداد عبر عدة كتل بيانات. في حالة لم تتناسب مدخلة بالكامل في كتلة واحدة، يدفع بها إلى كتلة البيانات التالية مع ضبط حقل rec_len في المدخلة السابقة بالشكل الصحيح

      ملاحظة: بما أن هذه القيمة لا يمكن أن تكون سالبة، عند حذف الملف ستعدل التسجيلة السابقة داخل الكتلة للإشارة إلى التسجيلة الصالحة التالية داخل الكتلة أو إلى نهاية الكتلة عندما لا توجد مدخلة دليل أخرى. في حالة حذف المدخلة الأولى داخل الكتلة، يتم إنشاء تسجيلة فارغة تشير إلى مدخلة الدليل التالية أو إلى نهاية الكتلة.
    - حقل طول الاسم name_len: قيمة 8 بت لا تحمل إشارة تشير إلى عدد بايتات بيانات المحارف في الاسم

هذه القيمة لا يجب أبدا أن تكون أكبر من 8 – rec_len. في حالة تحديث اسم مدخلة الدليل ولم يتناسب في مدخلة الدليل الموجودة، يمكن نقل المدخلة إلى مدخلة دليل جديدة بحجم كافي ويمكن أن تخزن في كتلة بيانات جديدة.

- حقل نوع الملف file_type: قيمة 8 بت لا تحمل إشارة للإشارة إلى نوع الكائن.

في المراجعة 0، هذا الحقل كان 8-بت العليا من حقل 16 بت name_len. بما أن جميع التطبيقات ما زالت تقيد أسماء الملفات إلى 255 محرف، هذه 8-بت كانت دائما 0.

هذه القيمة يجب أن تقابل نوع inode المحدد في مدخلة inode ذات الصلة.

- حقل اسم المدخلة name: (غالبا ISO-Latin-1) و لا يجب أن يكون أطول من 255 بايت بعد الترميز.

11. ^ عدد الكتل المنطقية المقروءة أو المكتوبة إلى القرص قبل الانتقال إلى القرص التالي. هذا يأثر على وضعية البيانات الوصفية في نظام الملفات وقد يسرع تخزين ريد RAID.
12. ^ عدد الكتل المنطقية المقروءة أو المكتوبة إلى القرص قبل العودة إلى القرص الحالي. يستخدمه مخصص الكتل إذا أمكن لخفض دورات read-modify-write في RAID5/6 (راجع Parity_bit#RAID)

قطع من البيانات المتتابعة التي تكتب أو تقرا من القرص قبل انتقال العملية إلى القرص التالي تسمى عادة : وحدات شريطية! stripe units أو strides أو Chunks بينما مجموعاتهم المنطقية logical groups التي تشكل عمليات شريطية واحدة تسمى شرائط strips أو stripes. كمية البيانات في الوحدة الشريطية الواحدة (stripe unit)، غالبا يعبر عنها بـ بايتات ويشار لها بأسماء عدة: حجم المجموعة chunk size، حجم الشريط stripe size, stripe depth، طول الشريط stripe length.

عدد أقراص البيانات في المصفوفة أحيانا يسمى عرض الشريط stripe width، لكن أيضا قد تشير إلى كمية البيانات داخل الشريط stripe.كمية البيانات في الشريط الواحد stride مضروب في عدد أقراص البيانات في المصفوفة (أي،stripe depth ضرب stripe width ; هذا في التمثيل الهندسية geometrical analogy سيعود بالمساحة) أحيانا يدعى حجم الشريط stripe size أو عرض الشريط stripe width. الشرائط العريضة أو الواسعة! Wide striping تحدث عندما توزع شرائط من البيانات (chunks) على مصفوفات متعددة، ربما جميع الأقراص في نظام الملفات

الشرائط الضيقة أو المحدودة! Narrow striping تحدث عندما شرائط من البيانات توزع على الأقراص في مصفوفة واحدة. (تنبيه: تأكد من هذه الترجمة و/أو راجع بقية النص في الموسوعة الحرة)

13. ^ أ، ب، ت، الحصص النسبة للقرص: تعني تقنين مساحة القرص أو عدد الملفات المنوحة للمستخدم والمجموعة على نظام الملفات. هذا يعني إمكانية منع أحد أو مجموعة المستخدمين من استهلاك كامل المساحة المتوفرة. وهناك نوعان من الحصص النسبية للقرص: الأول هو usage quota أو block quota ويحدد مساحة القرص المستخدمة، والثاني هو file quota أو inode quota، ويحدد عدد الملفات والأدلة التي يمكن إنشائها.ـ (راجع أكثر معلومات الموسوعة الحرة ووثائق نواة لينكس وموقع linuxquota).
14. ^ أ، ب، ت، ث حصص المشروع Project quota:
    - Project : المشروع هو تجميعه من inodes غير مرتبطة وقد تكون مبعثرة على أدلة مختلفة.
    - Project quota : نوع جديد من الحصص النسبية للقرص، الذي يكمل الأنواع الموجودة user/group quota. هذا النوع سوف يقنن المساحة المستخدمة من قبل المشروع!، بغض النظر عن المستخدم المنشئ للملفات في شجرة دليل المشروع. لكن مع هذا النوع لا يمكن استعمال group quotas على نفس نظام ملفات. وأنظمة الملفات التي تدعم Project quota هي: XFS و GPFS: "fileset" quota و Ext4 (جديد)، و Lustre! (قريبا)
    - هوية المشروع Project ID : inodes التي تنتمي إلى نفس المشروع، تملك هوية متماثلة، تماما مثل هوية المجموعة / المستخدم user/group ID.
15. ^ أ، ب، الفوقانية overhead : (الأعباء) موارد (عادة، في زمن المعالجة أو مساحة تخزين) مستهلكة لأغراض عرضية، لكنها ضرورية للغرض الأساسي. وفي سياق آخر تسمى تكاليف غير مباشرة.
16. ^ جدول توصيف مجموعات الكتل Block Group Descriptors Table : عبارة عن واصفات متتابعة تشكل جدول توصيف، كل مجموعة كتل تتضمن نسخة احتياطية من هذا الجدول مباشرة بعد كتلة Backup superblock ولأنها احتياطية، نظام الملفات يستخدم (في الاقلاع) فقط النسخ الأولية Primary superblock و primary group descriptor table الموجودة في مجموعة الكتل 0.

في المثال التالي سنعرض بعض مضمون النسخة الأولية من جدول توصيف مجموعات الكتل (في المجموعة 0)، ولأن الكتلة بحجم 4,096 بايت، superblock ستكون في الكتلة 0، وجدول توصيف المجموعات في الكتلة 1. (في حال كان حجم الكتلة 1,024 بايت، superblock ستكون في الكتلة 1 وجدول توصيف المجموعات في الكتلة 2) الخَرَج التالي يعرض اثنان من مدخلات توصيف المجموعات.

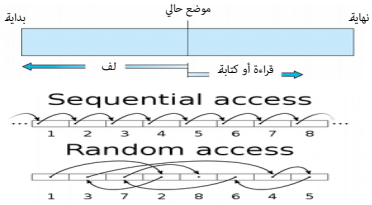
```
                # blkcat -f linux-ext3 ext3.dd 1 | xxd
                  0001 0203 0405 0607 0809 0A0B 0C0D 0E0F 0123456789ABCDEF
0000000: 0200 0000 0300 0000 0400 0000 d610 7b3f ..............{?
0000016: 0a00 0000 0000 0000 0000 0000 0000 0000 ................
0000032: 0280 0000 0380 0000 0480 0000 0000 8e3f ...............?
0000048: 0100 0000 0000 0000 0000 0000 0000 0000 ................
[REMOVED]
```

→ مدخلة المجموعة 0

→ مدخلة المجموعة 1

بايتات من 0 إلى 3 تشير إلى الكتلة 2 حيث تقع block bitmap بايتات من 4 إلى 7 تشير إلى الكتلة 3 حيث تقع inode bitmap بايتات من 8 إلى 11 تشير إلى الكتلة 4 حيث يقع inode table. في هذا المثال كانت 32,768 كتلة في كل مجموعة كتل، هذا يعني أن block bitmap تحتاج إلى 4,096 بايت، (أي كتلة كاملة) في هذا المثال كان أيضا 16,288 مؤشر فهرسة Inodes في كل مجموعة، إذن Inode bitmap تحتاج إلى 2,036 بايت ((16288 × 128) ÷1024). و Inode table يملك 16.288 مدخلة كل واحدة بحجم 128 بايت (حجم inode في ext3)، الحاصل سيكون 2,084,864 بايت. مع حجم الكتلة 4,096 بايت،Inode table يحتاج إلى 509 كتلة (2084864 ÷ 4096) ويمتد من الكتلة 4 إلى 512ـ ((512-4) +1) في الجدول مدخلة المجموعة 1 تبدأ عند الكتلة 32. كما نرى في بايتات من 32 إلى 35 أن Block bitmap في الكتلة 32,770 (0x8002) هذا بديهي لأننا نعلم أن المجموعة 1 ستبدأ في الكتلة 32,768 وأن النسخ من superblock و group descriptor table سوف تستخدم أول كتلتين. إذا كانت مجموعة الكتل لا تملك نسخة من superblock و Group Descriptors، المصفوفة الثنائية للكتل block bitmap ستقع في أول كتلة من المجموعة.

17. ^ بت التقييد! sticky bit هو بت في inode يشير إلى أن البرنامج التنفيذي executable program يجب أن يضل في الذاكرة بعد انتهاءه. في هذه الحالة فقط المستخدم الجذر أو مالك الملف يستطيع تعديل الملف، ونواة لينكس تتجاهل هذا بت في الملفات.في حالة تعيين sticky bit على الدليل، المستخدم الذي لا يملك الامتياز لا يستطيع حذف أو إعادة تسمية ملفات الأخرين في ذلك الدليل حتى وإن كان يملك حق الكتابة في الدليل. (راجع أكثر Chmod). بالمناسبة sticky bit في الدارات الرقمية له استخدام أخر.
18. ^ الأجهزة الحديثة مأمثلة للنفاذ المتتابع وتخفي قياساتها الفيزيائية عن نظام التشغيل. ولأن أنظمة الملفات الحديثة لم تعد تهتم كثيرا بتفاصيل القياسات الفيزيائية للقطاعات على القرص. كذلك الأقراص الحديثة لا تفشي الكثير من معلوماتها الفيزيائية. وكل ما نعرفه هو أن النفاذ المتتابع أسرع من النفاذ العشوائي أو المباشر. (لكن ما هي الدوافع الأخرى!؟ ربما الجواب عند ويكيليكس). قياسات القرص Disk geometry تعني تحديد أين يقع مكان القطاع X على القرص. بمعنى أخر، عدد الأسطوانات، عدد القطاعات لكل أسطوانة والرؤوس في القرص الثابت...إلى آخره.



النفاذ المتتابع Sequential access مثل ـ قراءة 100 بايت التالية. والنفاذ العشوائي أو النفاذ المباشر Direct access مثل: قراءة 10 بايت عند الإزاحة 300ـ

19. ^ بايتات التي تتبع الإزاحة 0x54 يجب أن تكون صالحة، وكلمة "ديناميكية"هنا تشير إلى inode بحجم متغير، والحجم الفعلي في بايت 0x58
20. ^ عموما، كتل البيانات التي تكتب إلى القرص من خلال قيد الحوادث تكتب حرفيا إلى ملف قيد الحوادث journal file بعد كتلة التوصيف descriptor block. لكن، إذا كانت 4 بايت الأولى من كتلة نظام الملفات متطابقة مع قيمة التوقيع في ترويسة jbd2 حينذاك 4 بايت تستبدل بأصفار ويتم تعيين علم "escaped" في لاحقة كتلة التوصيف descriptor block tag عند كتابتها إلى journal.
21. ^ أ، ب، ت هذه الحقول كانت سابقا على التوالي h_i_frag ـ h_i_fsize (8 بت) و m_i_frag أو m_i_fsize في Masix لكن بما أن (تقنية) fragments ليس مدعوم في لينكس وحنو هرد و Masix، رقم وحجم fragment.

دائما 0ـ وملغى في ext4. الآن حقل l_i_frag في ext4 مدمج في حقل l_i_fsize ويشكل 16 بت العليا من تعداد الكتل 48 بت للبيانات inode.

بالمناسبة: مصطلح Fragment مازال يستخدم في خرج dumpe2fs و tune2fs ؛ ويعني حجم الكتلة Fragment size وعدد الكتل لكل مجموعة Fragments per group

22. ^ تسجيل أو تسجيلة record تعني أيضا تركيب أو تركيبة struct أو بيانات مركبة compound data.

23. ^ الكتلة block هي أصغر وحدة تقيل العنونة، يحددها نظام التشغيل (هذه القيمة أس العدد اثنين، وستكون على الأقل بحجم قطاع على القرص (عادة 512 بايت)، أو حتى بحجم الصفحة الذاكرية). والكتلة المنطقية قد تكون أكبر من 512 بايت، على أقراص مثل، MO أو AD.

24. ^ أ، ب، ت، البذرة، البذرة العشوائية، أو القيمة الابتدائية random seed, seed state, seed هو عدد (أو متجه) يستعمل في بدأ مولد أعداد شبه عشوائية. (راجع الموسوعة)

25. ^ أ، ب، ت، HTree أو hashed B-Tree بنية بيانات شجرية مخصصة لفهرسة الأدلة تشبه بنية B-tree وكلاهما له عمق ثابت بمستوى واحد أو اثنين، مع معامل خرج عالي high fanout factor، وتستخدم قيمة هاش (مفاتيح) من اسم الملف، (بدلا عن اسم الملف أو اسم الدليل...الخ، المبحوث عنه) ولا تحتاج إلى شجرة بحث ثنائية متزنة ذاتيا (راجع الموسوعة).

- فهارس HTree indexes ext2 كانت من أجل ext2 لكن الرقعة patch لم تلحقها أبدا بالفرع الرسمي. وتمكين dir_index ممكن عند إنشاء ext2، لكن شفرة ext2 لن تعمل عليها.
- فهارس HTree indexes ext3 متوفرة في ext3 عن طريق تمكين ميزة dir_index..
- فهارس HTree indexes ext4 ستكون في حالة تمكين في ext4. هذه الميزة مطبقة منذ نواة لينكس 2.6.23ـ فهارس indices HTree تستخدم أيضا من أجل مديات extents عندما يحتاج الملف إلى أكثر من 4 مديات مخزنة في inode. في ext4، ميزة dir_index تسمح باستخدام مدخلات دليل شجرة الهاش.لكن يجب أن يملك inode الدليل المطابق تعيين INDEX_FL إذا استخدم.
- تنبيه: صفحة الموسوعة الحرة تقول أن HTree لا تحتاج إلى شجرة بحث ثنائية متزنة ذاتيا !

26. ^ أ، ب، ت، ملوثة! Dirty : مضمون بيانات يحتاج إلى إعادة كتابة إلى ذاكرة تخزين أكبر. أمثلة:

- البيانات الملوثة Dirty data، (تعرف أيضا rogue data)، هي بيانات غير دقيقةـ أو غير كاملةـ أو متضاربة. خصوصا في نظام الحاسوب أو قاعدة البيانات. البيانات الملوثة يمكن أن تتضمن أخطاء مثل أخطاء التهجئة، والترقيم، أو بيانات خاطئة مصاحبة للحقلـ أو بيانات غير كاملة أو قديمة، أو حتى بيانات مكررة في قاعدة البياناتـ يمكن مسحها عن طريق عملية تدعى تطهير البيانات data cleansing. (ما تبقى من النص في الموسوعة الحرة الانجليزية)
- قراءة القطاع من الصوان الملوث dirty buffer، الذي يعني حاجته إلى مزامنة (sync) الصوان الملوث dirty buffer، أولا.
- كتابات القرص: خابية الصفحات الذاكرية page cache (أحيانا تسمى disk cache) أيضا تساعد في الكتابة إلى القرص. الصفحات في الذاكرة الرئيسية RAM المعدلة أثناء كتابة البيانات إلى القرص سوف توسم بالملوثة "dirty" ويجب تخليصها (to flushed) إلى القرص قبل تحريرها.[...]
- بت ملوث! أو بت معدل،dirty bit, modified bit/ هو بت مقترن بكتلة في ذاكرة الحاسوب، يشير ما إذا كانت الكتلة الموافقة في الذاكرة معدلة أو لا.

27. ^ أ، ب، ت، ث، أعلام الميزات feature flags (أيضا feature toggle, feature switch, feature flipper, conditional feature...feature set...إلى آخره) تقنية تستخدم في تطوير البرمجيات تحاول تقديم بديل للصيانة تفرعات عدة من الشفرة الأصلية (تعرف بـ feature branches) مثال على ذلك، الميزة التي يمكن اختبارها حتى قبل اكتمالها أو جاهزيتها للإصدار.أعلام الميزات، تستخدم في إخفاء ، وتمكين أو تعطيل الميزة في زمن التشغيل مثلا، أثناء عملية التطوير المطور يمكنه تمكين الميزة من أجل الاختبار وتعطيلها على المستخدمين الآخرون. (راجع الموسوعة الحرة)

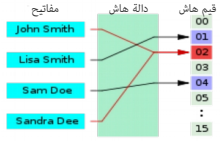
28. ^ أ، ب، تخصيص الكتل المتأخر Delayed Allocation أو التخصيص عند التخليص!ـ Allocate-on-flush، هذه التقنية تعني عند كتابة البيانات إلى القرص بدلا من تخصيصها/توزيعها فورا (على القرص)، سوف تخزن في الخابية cache. والبيانات في الخابية تكتب فقط بعد تخليص الخابية "flush"-ing the cache"، وهذه ستكون فرصة للمحصص الكتل لأمثلة عملية التخصيص باستخدام المديات. هذه الميزة تستخدمها أنظمة الملفات Btrf، ZFS، XFS، Reiser4، HFS+، ext4،ـ والميزة تشبه كثيرا تقنية قديمة في نظام بركيلي UFS (نظام ملفات يونكس) تدعى إعادة توزيع أو تخصيص الكتل"block reallocation" عندما يستلزم تخصيص الكتل لحفظ الكتابات المعلقة (في الانتظار)، يتم طرح مساحة القرص المخصصة للبيانات الملحقة appended من عداد المساحة الحرة free-space counter، ولا تخصص في الواقع في مصفوفة الثنائية للمساحة الحرة free-space bitmap. ولكن يحتفظ بالبيانات الملحقة في الذاكرة حتى يتم تخليصها بكتابتها (to flushed) إلى التخزين نتيجة الضغط على الذاكرةـ عندما تقرر النواة التخلص من بيانات الصوان الملوثة dirty buffers، أو عندما التطبيق ينفذ نداء النظام sync (المزامنة) في يونكس.(.. بقية النص في الموسوعة الحرة).

29. ^ أ، ب، ت، ث صورة زمنية انتقائية (للنظام)، لقطة، سناب شوت snapshot تدل على وضعية النظام في مرحلة زمنية معينة. وكلمة snapshot مستوحاة من التصوير الفوتوغرافيـ ويمكن أن تشير إلى نسخة فعلية (صورة) من وضع النظام أو قدرة capability توفرها أنظمة محددة.

30. ^ ATA Trim command (المعروف بـ تريم TRIM في مجموعة أوامر ATA، و UNMAP في مجموعة أوامر SCSI). عن طريق هذا ATA Trim. نظام التشغيل يخبر SSD عن كتل البيانات التي لم تعد مطلوبة أو مستخدمة ويمكن مسحها داخليا. وحتى يعمل ATA Trim، يجب أن يدعم قرص SSD ونظام التشغيل ونظام الملفات هذه الميزة. مثلا للتأكد من وجود هذه الميزة على القرص الأول:

hdparm -I /dev/sda | grep -i trim. ثم تمكينها مثال **tune2fs -o discard /dev/sda1** . ريد هات RedHat (وهي شركة استغلالية للمبرمجين وللأنظمة المفتوحة!) لا توصي باستخدام مستويات ريد الرمحية 1، 4، 5، و 6 على أقراص SSD، مع معظم تقنيات ريد، لأن أثناء التهيئةـ معظم وسائل إدارة ريد RAID (مثل، mdadm في لينكس) تكتب إلى جميع الكتل على الأقراص لتأكد من عمل تدقيق المجاميع بشكل صحيح، وهذا يجعل SSD يعتقد أن جميع الكتل خارج المنطقة الإضافية spare area مستخدمة. ولهذا تأثير سلبي على الأداء، شركة ريد هات توصي باستخدام RAID 1 أو RAID 10 من أجل RAIDs LVM على SSD، لأن تلك المستويات تدعم TRIM (أي discard في مصطلحات لينكس)، أدوات LVM لا تكتب إلى جميع الكتل عند إنشاء الوحدة RAID 1 أو RAID 10

31. ^ أ، ب، ت، ث، ج، ح، خ، د دالة تجزئة أو دالة هاش hash function هي أية خوارزمية أو دالة رياضية تُحوّل (أو ترمز) مجموعة كبيرة من البيانات بأي حجم إلى بيانات بحجم أصغر وثابت، تستخدم في جداول هاشـ وعلم التعمية (علم التشفير). وهي عادةً ما تكون عدد صحيح يعمل بمثابة مؤشر لمجموعة من البيانات. وتسمى القيم التي تعود بها دالة هاش: قيم هاش hash values أو رموز هاش hash codes أو مجاميع هاش digests أو فقط تسمى هاش hashes.. تُستخدم دالات هاش غالباً لتطوير الجداول أو مهام البيانات مثل: العثور على العناصر الموجودة داخل قاعدة البيانات، والكشف عن صفوف مماثلة في ملف كبير، وإيجاد مساحات مماثلة في تسلسلات دي إن إيه DNA، وغيرها. وقد تحدد دالة هاش مفتاح أو اثنين من مفاتيح قيمة هاش نفسها. وفي كثير من التطبيقات، يجب تقليل نسبة التصادم Collision. وهذا يعني أنه يجب على دالة هاش رسم خريطة لمفاتيح قيم هاش بالتساوي قدر الإمكان. وقد تتطلب بعض التطبيقات خصائص أخرى. وعلى الرغم من أن الفكرة نشأت في الخمسينيات، لا يزال موضوع تصميم دالات هاش قيد البحث. الآن دالات هاش ترتبط بتدقيق المجموعـ وتدقيق الأرقامـ والبصماتـ والدالات العشوائيةـ والضغط مع فقدان معلومات (ضغط فقود) ورموز تصحيح الخطأـ وشفرة الكتابةـ ودالة هاش الرمزيةـ ورغم تداخل هذه المفاهيم إلى حد ما، لكل مفهوم استخداماته واحتياجاته الخاصة. كما يتم تصميم كل واحدة منهم بشكل مختلف.



دالة هاش تربط (maps) الأسماء بالأعداد الصحيحة من 0 إلى 15.

لاحظ هنا تصادم collision بين مفاتيح "John Smith" و "Sandra Dee"

إذن، رموز أو شفرة الهاش hash code، هي شفرة code أو قيمة value مولدة بواسطة دالة تجزئة أو دالة هاش hash function من أجل تمثيل قطعة من البيانات. أما فعل هاش فهو التحويل وفقا للدالة التجزئةـ والأفعال المشتقة هي hash (to)، rehash، hash. أما hash map أو hashmap فتعني جدول الهاش hash table أو hashtable وجدول هاش هو أحد بنى المعطيات في علم الحاسوب يملك خصائص المصفوفات الترابطية associative array (يطبق كمتجه vector، أي موقع في الذاكرة) ويمكن باستخدامه إسناد (بتطبيق دالة التجزئة hash function) قيمة إلى مفتاح ما في ذاكرة الحاسب. والبحث عن قيم محددة بسرعة كبيرة مقارنة ببنى المعطيات الأخرى.

32. ^ أ، ب، ت، ث، مصفوفة ثنائية، خارطة ثنائية Bitmap, Bit map, Bit array هي بنية بيانات مصفوفية، تخزن بتات بشكل متراص. مثلا قد تكون في شكل ملف يتعقب المساحة المستخدمة والغير مستخدمة. بحيث كل بت يمثل كتلة تقبل العنونة 1 أو 0. وتعبير bitmap يستخدم أكثر في حالات، مثل تخصيص الصفحات الذاكريةـ و inodes، وقطاعات القرص... إلى آخره.

يستخدم أيضا تعبير bitmap (خريطة نقطية) لإشارة إلى الصور التسامتية (الرسوميات النقطية)، التي يمكن أن تستخدم عدة بتات في كل بكسل (العمق اللوني)

33. ^ أ، ب، ت، ث، ج، ح، يونكس لا يشترط أية بنية داخلية للملف العادية في نظام الملفات. من وجهة نظر نظام التشغيل، هناك فقط نوع ملف واحد. وبالتالي البنية والتأويل يعتمدان كليا على كيفية تفسير البرمجية للملف. رغم ذلك، نظام يونكس يملك بعض الملفات الخاصة. التي يمكن تمييزها بواسطة الأمر stat أو ls -l الذي يعرض نوع الملف في أول حرف أبجدي في حقل أذون نظام الملفات.

- الملف الاعتيادي Regular file: الملفات في الحاسوب تسمى كذلك ملفات اعتيادية لتمييزها عن الملفات الخاصة. وتظهر في بداية خرج ls -l بالشكل '-' بدون محرف مخصص في حقل النمط مثال:

```
# ls -l /etc/passwd
-rw-r--r-- ... /etc/passwd
```

- الدليل Directory : هو الملف الخاص الأكثر شيوع على الحاسوب، تخطيط ملف الدليل يحدده نظام الملفات المستخدم. ولكثرة أنظمة الملفات، الأصيلة! والغير أصيلة، المتوفرة تحت يونكس، لن تجد تخطيط موحد لملف الدليل (دليل). أيضا، حذف الملف يعني إزالة الملف من الدليل، ولا يعني حذف المضمون من القرص. الدليل موسوم بمحرف d كأول حرف في حقل النمط في خرج ls -dl مثال:

```
# ls -dl /
drwxr-xr-x 26 root root 4096 Sep 22 09:29 /
```

- وصلات رمزية / وصلات لينة symbolic link, soft link, symlink : هي مرجع إلى ملف أخر، والوصلة ملف خاص يخزن كتمثيل نصي لمسار الملف في المرجع (وهذا يعني أن الوجهة يمكن أن تكون مسار نسبي، أو غير موجودة إطلاقا) الوصلة الرمزية موسومة بالمحرف l (أي حرف L) كأول حرف من سلسلة النمط. مثال:

```
lrwxrwxrwx ... termcap -> /usr/share/misc/termcap
```

- أنبوبة اتصال مسماة Named pipe / FIFO : إحدى نقاط قوة يونكس كانت دائما في ميزة تواصل أو تبادل البيانات بين العمليات. ومن بين الوسائل التي بوفرها OS نجد الأنابيب التي تصل خرج عملية بدخل عملية أخرى. هذا جيد إذا كانت كلتا العمليتين موجودة في نفس محال العملية الأم، التي بدأها نفس المستخدم. لكن، هناك حالات حيث العمليات الموصولة يجب أن تستخدم أنابيب اتصال مسماة إحدى هذه الحالات تقع عندما يجب تنفيذ العمليات تحت أذون وأسماء مختلفة للمستخدمين. أنابيب الاتصال المسماة هي أيضا ملفات خاصة وقد تتواجد في أي مكان على نظام الملفات. الملفات الخاصة لأنابيب الاتصال المسماة تنشأ بالأمر mkfifo مثال mkfifo mypipe أنبوبة الاتصال المسماة موسومة بحرف p كأول حرف من سلسلة النمط. مثال:

```
prw-rw---- ... mypipe
```

- مقبس Socket (مقبس مجال يونكس!، مقبس نطاق يونكس!): هذا أيضا ملف خاص يستخدم في تبادل البيانات بين العمليات. ويمكن عمليتين من التواصل، إلى جانب إرسال البيانات، العمليات يمكنها أيضا إرسال توصيف الملفات عبر اتصال مقبس مجال يونكس! باستخدام نظام sendmsg() و recvmsg() وعلى خلاف، أنابيب الاتصال المسماة التي تسمح فقط بتدفق للبيانات أحادي الاتجاه، المقابس لها قدرة على التواصل في كلا الاتجاهين. المقبس موسوم بالمحرف s كأول حرف من سلسلة النمط. مثال:

```
srwxrwxrwx /tmp/.X11-unix/X0
```

- الجهاز المحرفي والجهاز الكتلي (ملف الجهاز) Device file: في يونكس، كل شيء تقريبا هو ملف وله موقع في نظام الملفات؛ بما في ذلك أجهزة مثل القرص الثابت. الاستثناء البارز للأجهزة والملفات التي تمثلها سيكون في أجهزة الشبكة التي لا تظهر في نظام الملفات ولكن تعامل على حدة. ملفات الأجهزة تستخدم في تطبيق حقوق النفاذ (أذون) وتوجيه العمليات على الملفات إلى مشغلات الأجهزة المناسبة. يونكس يميز بين الأجهزة المحرفية والأجهزة الكتلية، وتقريبا الفرق سيكون كالتالي:
  - الأجهزة المحرفية (أحيانا تسمى raw devices) تشترط فقط تدفق متسلسل (بايتات) من دَخْل أو تقبل تدفق متسلسل من خَرْج (مثل لوحة المفاتيح)؛
  - الأجهزة الكتلية (كتل) يمكن النفاذ إليها بشكل عشوائي (مثل القرص الثابت، و CD-ROM)؛

مع ذلك، أقسام القرص مثلا قد تملك النوعين؛ الأجهزة المحرفية مع نفاذ عشوائي بدون ذاكرة وسيطة إلى الكتل على القسم والأجهزة الكتلية مع نفاذ عشوائي ذو ذاكرة وسيطة إلى الكتل على القسم. الجهاز المحرفي موسوم بالمحرف c كأول حرف من سلسلة النمط. وبنفس الأسلوب، الجهاز الكتلي موسوم بالمحرف b. في لينكس، الأجهزة يمكن النفاذ إليها عن طريق هذه الملفات الخاصة، التي عادة تتواجد تحت دليل /dev. مثال:

```
crw------- ... /dev/null
brw-rw---- ... /dev/sda
```

- باب Door ! : ملف خاص لتبادل البيانات بين العمليات بين العميل والخادوم، حاليا هذا النوع مطبق فقط في سولاريس. الباب! موسوم بالمحرف D (كبير) كأول حرف من سلسلة النمط. مثال:

```
Dr--r--r-- ... name_service_door
```

34. ^ في يونكس ومكتبة سي، يستخدم محرف لاشيء أو رمز لاشيء null character لإنهاء السلاسل النصية strings؛ text و null-terminated strings قد تعني ASCIZ أو ASCIIZ حيث Z يرمز للصفر

35. ^ أ، ب، ترميز مفتوح! Open-coding (يعرف أيضا بالتضمين inlining): مثلا في SBCL تعني استبدال نداءات الدالة (أو استدعاءات الوظيفة) function calls بالتجميع الداخلي inline assembly

36. ^ كائن ثنائي ضخم Binary large object أو BLOB هو ملف كبير جدا، يتطلب معاملة أو معالجة خاصة نتيجة لحجمه (في الإرسال، التخزين، التنزيل، إلى آخره) في البرمجيات الحرة، Binary blob ملف غرضي غير حر non-free object file يحمل في النواة (راجع LKM) وأحيانا يطبق على الشفرة خارج النواة مثل برامج مساحة المستخدم أو صور البرنامج الثابت. مصطلح blob استخدم أول مرة في نظام إدارة قاعدة البيانات :وكلمات مثل كائن ثنائي، كائن ثنائي ضخم BLOB Binary objects, BO, binary large object كلها تشير إلى مجموعة من الملفات الرقمية التي تمثل البيانات الثنائية. مثل الصور والوسائط الأخرى

37. ^ تمكين الاستيثاق والحفاظ على التكامل الشفاف للملفات في وضعية القراءة فقط transparent integrity protection and authentication of read-only files.
الميزة fs-verity تستخدم شجرات ميركل Merkle tree (التي تسمى أيضا hash tree) لإضافة تأدية وظيفية تشبه آلية dm-verity على مستوى الملفات per-file basis. هذه الميزة تعمل على التحقق من صحة أية كتلة في الملف زمن log(filesize). تمكين الميزة على الملف يتطلب تنفيذ tune2fs -O verity أو mkfs.ext4 -O verity عند تهيئة نظام الملفات.

38. ^ أ، ب، تعداد المراجع Reference count، عدة أنظمة ملفات تملك تعداد مراجع إلى الكتلة أو الملف، مثل تعداد روابط inode في يونكس.في هذه الأخيرة عندما ينزل التعداد إلى الصفر يمكن إلغاء تخصيص الملف بأمان. بعض أنظمة ملفات يونكس إلى جانب المراجع من الأدلة، تسمح أيضا بالمراجع من العمليات الحية. ومع ملفات قد لا تكون متواجدة في هرمية نظام الملفات.

- في حقل تعداد المراجع h_refcount قيمة 32 بت تشير إلى عدد الملفات التي تستخدم هذه الكتلة لأن الملف الذي يملك نفس الخصائص الممتدة سيتشارك في كتلة الخصائص الممتدة. التَعْداد المرجعي. يزداد في كل مرة يتم فيها إنشاء رابط إلى كتلة الخصائص أو ينقص عند إزالة الرابط. وعند نزول القيمة إلى 0، تحرر كتلة الخصائص.

39. ^ صفر / 0 يستخدم كقيمة حارسة! sentinel value للإشارة إلى العدم NULL أو الإشارة إلى inode غير موجود. هذا يشبه مؤشرات NULL في C. بدون sentinel ستكون هناك حاجة إلى بت إضافي للتأكد من حالة تعيين inode في التركيبة struct. جميع عناوين inodes والكتل تبدأ مع الواحد 1، أول كتلة على القرص ستكون الكتلة 1، و 0 يستخدم للإشارة إلى عدم وجود كتلة.(الملفات ذات الفراغات Sparse files يمكن أن تملك هذه داخليا) على سبيل المثال، في أنظمة الملفات القديمة، حيث الأدلة تمثل بواسطة مصفوفة ثابتة من مدخلات الملفات، عند حذف ملف ينتج عن ذلك تعيين قيمة inode إلى 0 في المدخلة، وعند البحث المعمق في الدليل، أي مدخلة تملك 0 inode سوف يتم تجاهلها.
هذه بعض الأمثلة العامة التي تستخدم القيم الحارسة ! sentinel values :

- محرف / رمز لاشيء : للإشارة إلى نهاية سلسلة null-terminated string
- مؤشّر للاشيء : للإشارة إلى نهاية قائمة متصلة أو شجرة.
- عدد صحيح سالب للإشارة إلى نهاية متتالية من أعداد صحيحة غير سالبة

40. ^ أ، ب، ت، ث، ج، فهرس/ مؤشر index هو عدد صحيح أو مفتاح أخر يشير إلى موقع بيانات. على سبيل المثال، داخل مصفوفة، متجه، جدول قاعدة بيانات، مصفوفة ترابطية، أو جدول هاش.

41. ^ يمكن اعتبار القرص كمصفوفة كتل، كل كتلة تتضمن قطاع واحد أو اكثر، القطاع عادة بحجم 512 بايت، أو أكبر في بعض الأقراص الحديثة. القطاعات يمكن النفاذ إليها بشكل عشوائي، وقراءة وكتابة القطاعات ستكون ذرية. القطاع مفهوم يستخدم على مستوى القرص، والكتلة مفهوم يستخدم على مستوى نظام الملفات، مثلا، في لينكس عادة كل كتلة قرص 4 كيلوبايت (الموافق للصفحة الذاكرية)

42. ^ صيغة الدليل المفهرس Indexed Directory : استخدام الصيغة المعيارية للدليل القائمة المتصلة سيكون بطيء جدا بنمو عدد الملفات. ولذلك، لتحسين الأداء في مثل هذه الأنظمة، يستخدم فهرس هاش hashed index الذي يسمح بتحديد موقع الملف المطلوب بسرعة. في حالة استخدام صيغة الدليل المفهرس، سيتم تعيين بت INDEX_FL في حقل i_flags في inode للدليل

للتوافق خلفيا مع التطبيقات الأقدم، الدليل المفهرس indexed directory يحتفظ أيضا بصيغة الدليل المتصلة linked directory. في حال كان هناك أي تعارض بينهما ستكون الأفضلية للأدلة المتصلة. والتوافق الخلفي يتحقق بوضع مدخلات دليل مزيفة بداية الكتلة 0 من كتل بيانات الدليل المفهرس. هذه المدخلات المزيفة جزء من dx_root وتستضيف معلومات الدليل المتصلة للمدخلات "." و "..".

مباشرة بعد بنية جذر الدليل المفهرس Indexed Directory Root ستكون هناك مصفوفة من مدخلات الدليل المفهرس Indexed Directory Entry التي تصل إلى نهاية كتلة البيانات أو حتى تتم فهرسة جميع الملفات. عندما يتجاوز عدد الملفات التي ستفهرس عدد مدخلات الدليل المفهرس التي يمكن أن تتناسب في كتلة (حقل Limit في Indexed Directory Entry)، يتم إنشاء مستوى من الفهارس الغير مباشرة. و indirect index هو كتلة بيانات أخرى مخصصة من أجل inode للدليل الذي يتضمن مدخلات الدليل.

43. ^ أ، ب، Indexed Directory Entry: مدخلات الدليل المفهرس من أجل البحث السريع عن رقم inode المرتبط بالهاش من اسم الملف. هذه المدخلات تقع مباشرة بعد المدخلة المزيفة للدليل المتصلة في كتل بيانات الدليل، أو مباشرة بعد جذر الدليل المفهرس Indexed Directory Root. وأول مدخلة في الدليل المفهرس بدلا من أن تتضمن الهاش الفعلي ورقم الكتلة، ستتضمن العدد الأقصى للمدخلات الدليل المفهرس التي يمكن أن تتناسب في الكتلة والعدد الفعلي للمدخلات الدليل المفهرس المخزن في الكتلة. تفاصيل صيغة هذه المدخلة الخاصة ستكون في بنية Count و Limit (في Indexed Directory Entry) ومدخلات الدليل الأخرى مرتبة حسب قيمة الهاش من أصغر قيمة عددية إلى أكبرها.

44. ^ قيمة 32 بت تشير إلى عدد الكتل المستخدم حاليا من قبل الخصائص الممتدة. في لينكس قيمة h_blocks الأكبر من 1 تعتبر غير صالحة. حاليا لينكس لا يدعم لوائح الخصائص مع أكثر من كتلة واحدة، (راجع XATTR(7)) لكن أنظمة أخرى قد تدعم ذلك. هذا فعليا يقيد كمية الخصائص الممتدة إلى ما يمكن أن يتناسب في كتلة واحدة. يبدوا أن دعم الخصائص الممتدة لا يوجد في ext2 على جنو هيرد (نواة)

45. ^ أ، ب، ت، ث، المالك والمجموعة (المستخدمين) في inode : في معظم التطبيقات، المالك والمجموعة بقيم 16 بت، لكن في بعض تطبيقات هيرد ولينكس الحديثة معرف المجموعة والمالك سيكون 32 بت.عند استخدام قيم 16 بت، فقط الجزء المنخفض سيكون صالح، بينما في القيمة 32 بت كلا الجزأين العلوي والمنخفض سيستخدم. مع إزاحة الجزء العلوي إلى اليسار 16 خانة ثم إضافته إلى المنخفض.

الجزء المنخفض من قيمة المالك والمجموعة يقع على التوالي في ext4_inode.i_uid و ext4_inode.i_gid

الجزء العلوي من قيمة المالك والمجموعة في هيرد يقع على التوالي في ext4_inode.osd2.hurd.h_i_uid_high و ext4_inode.osd2.hurd.h_i_gid_high

الجزء العلوي من قيمة المالك والمجموعة في لينكس يقع على التوالي في ext4_inode.osd2.linux.l_i_uid_high و ext4_inode.osd2.linux.l_i_gid_high

46. ^ أ، ب، الذرية هي حالة نظام ما (غالبا نظام قاعدة بيانات) حيث يشترط إما أن تكون جميع المراحل مكتملة أو غير مكتملة. أي لا مكان للحلول الوسط. الذرية تعني أن تعاملات قاعدة البيانات إما أن تنفذ جميع عملياتها بشكل كامل، أو لا ينفذ أي منها. مثال آخر: العملية الذرية أو الموحدة Atomic operation) هو مفهوم في الحوسبة المتوازية يعني أن العملية ككل لا يمكن تقسيمها أو مقاطعتها أي أنه يمكن التعامل معها كوحدة واحدة من قبل بقية النظام. كون العملية ذرية يعطي ضمانات قوية لباقي البرنامج بأن العملية ككل ليس لها إلا ناتجين محتملين. إما النجاح وإما الفشل. في حالة مقاطعتها لن يؤدي ذلك إلى وصول البرنامج لحالة شاذة. أي أن المقصود بموحد ليس أنها ليست عرضة للتقسيم ولكن أن سير العملية لن يتأثر بحدوث تقسيم لها من عدمه. وبالتالي لا يكون المبرمج مضطرا إلى تحليل سلوك أجزاء العملية في حالة المقاطعة. تعتبر الأوامر الموجهة للمعالج عمليات ذرية. في حالة احتياج البرنامج لعملية ذرية تشمل عدة أوامر متتالية يتم استخدام مفهوم استبعاد التشارك سواء على مستوى المعالج أو البرمجية. (الموسوعة الحرة -- العربية)

47. ^ عند إنشاء نظام ملفات ext4، مناطق جداول inode tables الموجودة يجب مسحها (بالكتابة الفوقية بمحارف nulls، أو "التصفير"). الميزة lazyinit (lazy initialization) ينبغي أن تسرع في عملية إنشاء نظام الملفات، لأنها لا تهيئ فورا جميع جداول inode tables، وسوف تهيئهم عوض ذلك تدريجيا أثناء عملية الوصل الابتدائية (للنظام الملفات) في الخلفية. راجع أكثر صفحة man لأداة MKE2FS مع الخيارات التالية:

lazy_itable_init[= <0 to disable, 1 to enable>] , lazy_journal_init[= <0 to disable , 1 to enable>] , uninit_bg

في الوصل الأول lazyinit ستكتب الكثير من المعلومات إلى القرص، في بادئ الأمر ext4lazyinit (عملية في النواة) ستكتب ما يصل إلى 16,000kB/s إلى الجهاز، أي تستخدم قدر كبير من bandwidth (معدل نقل البيانات) على القرص.

48. ^ التدقيق الدوري عن الأخطاء في البيانات الوصفية، نظام ملفات ext4 :
    - واصف أو توصيف مجموعة الكتل سيكون محمي بواسطة التدقيق الدوري عن الأخطاء CRC16.
    - على أنظمة ملفات 64-بت، يمكن تمديد الحقل إلى 32-بت، أو حشو (تقليص) 32-بت crc في 16 بت (وفقا لمعلومات فصل"CRC Stuffing" في موقع Ext4_Metadata_Checksums).
    - قيد الحوادث: jbd2 تستخدم journal_checksum للتأكد من تكامل journal وتدعم تدقيق مجاميع CRC32، MD5، أو SHA1، (منذ لينكس 3.0) يبدوا أنها تدعم فقط CRC32. الذي يمكن تغييره بسهولة إلى CRC32c.

49. ^ عند إنشاء دليل، تعداد الروابط link count سيبدأ من 2 ، هذا يعني رابط للدليل الحالي نفسه، وآخر للدليل الأم. (بالإضافة إلى الروابط الفرعية، إن وجدت) وصحيح أن الدليل ملف (خاص) يتضمن لائحة من أسماء الملفات، لكن (باستثناء "." و "..") وجود روابط صلبة متعددة إلى الدليل، سينشأ عنه حلقات داخل بنية الأدلة، بدلا من بنية متفرعة كالشجرة. ولهذا السبب، إنشاء روابط صلبة إلى الأدلة غالبا ممنوع في أنظمة لينكس، حتى وإن كان ممكن نظريا.

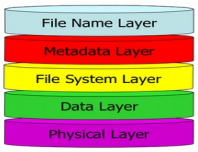
50. ^ أ، ب، ت، ث، ج، الكتل المحجوزة لنمو نظام الملفات مستقبلا GDT/GDG : بما أن ext4 يمكن أن يتضخم مستقبلا يتم حجز هذه الكتل عند إنشاء نظام الملفات أول مرة، التوسع أو إعادة تحجيم نظام الملفات في وضع متصل on-line resizing باستخدام أداة resize2fs، يعني المزيد من الكتل والمزيد من مجموعات الكتل. وبالتالي الحاجة للمزيد من هياكل Group Descriptors.

قيمة GDT ستكون إما صفر أو عدة كتل ودائما تتبع Group Descriptors. ويمكن تغيير القيمة باستخدام خيار resize=max-online-resize. في أداة mke2fs. مبدئيا، يمكن مضاعفة حجم نظام الملفات بمقدار 1024x مرة. أيضا يمكن إعادة تحجيم نظام الملفات في وضع غير متصل off-line عن طريق أداة resize2fs، إذا كان لا يملك تعيين ميزة resize_inode (المفيدة في أنظمة LVM). تعيين ميزة COMPAT_RESIZE_INODE يتطلب أيضا تمكين ميزة RO_COMPAT_SPARSE_SUPER أو COMPAT_SPARSE_SUPER2 ويعني وجود هذه الكتل المحجوزة

51. ^ في نظام ملفات ext4 المواقع على القرص تحدد بواسطة الكتل المنطقية، وليس كتل LBAs، وحجم كتل نظام الملفات 4 كيلوبايت (الوحدة التقريبية على مستوى نظام الملفات) وهو نفس حجم الصفحات (في الذاكرة) على أنظمة x86 وحجم الكتلة الافتراضي لطبقة الكتل (block layer)، رغم ذلك حساب الحجم الفعلي سيكون 2 ^ (10 + sb.s_log_block_size) بايت.
    - سابقا عندما كانت سعة التخزين صغيرة، والأقراص الثابت بطيئة كان الحجم يساوي: logical sector = physical sector = IO block = 512 bytes
    - حاليا، مع سعة التخزين الكبيرة، في أقراص SSD أو HDD السريعة أصبح الحجم يساوي : logical sector = physical sector = 512 bytes, IO block = 4096
    - IO block هي سعة نقل البيانات بين الجهاز drive ونظام التشغيل OS، وهي ناتج ضرب حجم القطاع الفيزيائي/المنطقي (بمعنى وحدة قياس للعمليات I/O operations لأن معظم أنظمة الملفات ترتكز على جهاز الكتل. وهو مستوى تحريدي). راجع أيضا خيارات MKE2FS خصوصا b- block-size و T- usage-type

52. ^ يمكن تعطيل أو تمكين الميزة باستخدام أداة tune2fs مع خيار feature[^] O-أو عند إنشاء نظام الملفات: mke2fs -t ext4 -O ^sparse_super /dev/sda1

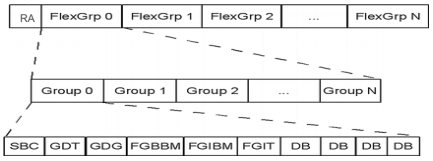
53. ^ الطبقة أحد عناصر الهرمية، مستوى تجريدي أو طبقة تجريدية abstraction layer / abstraction level: طريقة لإخفاء التفاصيل التطبيقية لمجموعة وظيفية محددة (أو ذات تأدية وظيفية محددة).



نموذج تصوري للنظام الملفات من5 طبقات

54. ^ غياب ميزة، huge_file يعني أن حجم الملف أقصاه سيكون 2^{32} كتلة منطقية. في حالة تعيين هذه الميزة، بدون تعيين HUGE_FILE_FL في inode. سيعني إمكانية وجود حجم ملف 2^{64} كتلة منطقية (حجم كبير جدا) وفي حالة تعيين أيضا HUGE_FILE_FL حجم الملف أقصاه سيكون 2^{64} كتلة نظام الملفات.

55. ^ أنواع مجموعات الكتل داخل مجموعات الكتل المرنة Flexible Block Groups (شرح مع أمثلة) لكن قبل ذلك هذا تذكير بالمخطط:



SBC	Super Block Copy	نسخة من الكتلة العليا (بيانات وصفية عامة للنظام الملفات)
GDT	Group Descriptor Table	جدول توصيف مجموعات الكتل
GDG	Group Descriptor Growth Blocks	الكتل المحجوزة لنمو هياكل توصيف المجموعات Group Descriptor
FGBBM	Flex Group Block Bitmap	المصفوفة الثنائية للكتل في مجموعة الكتل المرنة
FGIBM	Flex Group Inode Bitmap	المصفوفة الثنائية للمؤشرات الفهرسة في مجموعة الكتل المرنة
FGIT	Flex Group Inode Table	جدول مؤشرات الفهرسة في مجموعة الكتل المرنة
DB	Data Block	كتلة بيانات (متعددة)
RA	Reserved Area	منطقة محجوزة (حشو قبل المجموعة 0)

في أنظمة الملفات التي تستخدم حجم الكتلة 1 كيلوبايت، الكتلة 0 ليست جزء من مجموعة الكتل 0 block group.

على أية حال، في المثال التالي النظام كان يملك 120 مجموعة ولأن خرج (ملف) سيكون طويل سوف نعرض بعضها فقط ومن كل نوع : **dumpfile** < **dumpe2fs /dev/sda1**

مجموعة الكتل 0 (الأولى) Block Group 0 وتتضمن:

- 0–32767 كتلة، لأن كل مجموعة كتل تتضمن 32 كيلوبايت (32768) كتلة
- الكتلة العليا الأولية Primary superblock عند الكتلة 0.
- جدول توصيف المجموعات Group Descriptors عند الكتلة 1 (كتلة واحدة فقط)
- الكتل المحجوزة Reserved GDT Blocks، من الكتلة 2 إلى 955.
- المصفوفة الثنائية للكتل Block Bitmap عند الكتلة 956، والمصفوفة الثنائية للمؤشرات الفهرسة Inode Bitmap عند 972. (ستعرف فيما بعد لماذا Inode Bitmap ليست عند 957)
- جدول مؤشرات الفهرسة inode table من الكتلة 988 إلى 1496. إذن هناك 1496 − 988 +1 = 509 كتلة. كما تشير إليها Superblock مع إضافة +1 لأن 1496 ستكون **inclusive**.

```
Group 0: (Blocks 0-32767) [ITABLE_ZEROED]
  Checksum 0x9a88, unused inodes 8131
  Primary superblock at 0, Group descriptors at 1-1
  Reserved GDT blocks at 2-955
  Block bitmap at 956 (+956), Inode bitmap at 972 (+972)
  Inode table at 988-1496 (+988)
  23630 free blocks, 8133 free inodes, 2 directories, 8133 unused inodes
  Free blocks: 9138-32767
  Free inodes: 12-8144
```

يمكن إيجاد الفهارس مباشرة بدون استخدام dumpe2fs، كتل Group Descriptors تبدأ من الكتلة الثانية، وأول ثلاثة قيم 32-بت ستكون مواقع البيانات الوصفية التالية Block Bitmap، و inode Bitmap، و inode Table.

```
# dd if=/dev/sda1 bs=4096 skip=1 count=1 status=none | hexdump -n 12 -s 0 -e '"%d %d %d\n"'   ( = 956 972 988)
```

مجموعة الكتل 1 (الثانية) Block Group 1 وتتضمن:

- نسخة من Superblock، لأن 1 قوة العدد 3 ($3^0 = 1$) ولأنها تملك نسخة من Superblock تملك أيضا نسخة من Group Descriptors وكتل Reserved GDT blocks

ملاحظة: هذه المجموعة لا تملك Block Bitmaps و inode Bitmaps و inode Tables. لكنها تشير إلى Block Group 0 (في المثال، Block Bitmap في Block Group 1 عند 957 + #0 bg) لأن المجموعة 0 و 1 في نفس Flexible Block Group. نفس الشيء مع inode Bitmap و inode Table. وهذا هو السبب لماذا Block Bitmap و inode Bitmap ليست متتابعة في Block Group 0، فالأولى تبدأ عند 956، والثانية عند 972. وبحجم 16 كتلة لأنها تتضمن معلومات جميع مجموعات الكتل الأخرى (من 1 إلى 15) في Flexible Block Group. نفس الشيء مع inode Table، الذي يملك 509 كتلة تشير إليها Superblock، من الكتلة 988 إلى 9132، تتضمن inode Tables للمجموعات من 1 إلى 15.

```
Group 1: (Blocks 32768-65535) [INODE_UNINIT, ITABLE_ZEROED]
  Checksum 0x5017, unused inodes 8144
  Backup superblock at 32768, Group descriptors at 32769-32769
  Reserved GDT blocks at 32770-33723
  Block bitmap at 957 (bg #0 + 957), Inode bitmap at 973 (bg #0 + 973)
  Inode table at 1497-2005 (bg #0 + 1497)
  31809 free blocks, 8144 free inodes, 0 directories, 8144 unused inodes
  Free blocks: 33726-33791, 33793-65535
  Free inodes: 8145-16288
```

ملاحظة: قد تستغرب لماذا inode Table في المجموعة 0 Block Group ينتهي عند 9132 وكتل حرة تبدأ عند 9138 ؟ ((509 × 16) = 8144 + (988) = 9132) في الواقع، الكتل بين 9132-9137 كتل بيانات مستخدمة، وهذا ليس بالغريب. إذا أعدنا الطرح، قد تحصل على خرج مختلف، لأن خرج كان بعد التلاعب بنظام الملفات. إذن، نفهم من هذا : ليس صحيح أن كتل البيانات تتبع هياكل البيانات الوصفية هذه. وصحيح هو أن جميع الكتل هي كتل بيانات، لكن بعضها يستخدم من أجل البيانات الوصفية. الكتل المستخدمة من أجل البيانات الوصفية توسع أيضا بالمستخدمة في Data Blocks Bitmap.

مجموعة الكتل 2 (الثلاثة) Block Group 2، وتتضمن :

- كتل بيانات فقط، المصفوفات الثنائية Data Block Bitmap و inode Bitmap، وجدول inode Table لهذه المجموعة موجودة في Block Group 0. تماما مثل Block Group 1.

```
Group 2: (Blocks 65536-98303) [INODE_UNINIT, BLOCK_UNINIT, ITABLE_ZEROED]
  Checksum 0xeabf, unused inodes 8144
  Block bitmap at 958 (bg #0 + 958), Inode bitmap at 974 (bg #0 + 974)
  Inode table at 2006-2514 (bg #0 + 2006)
  32768 free blocks, 8144 free inodes, 0 directories, 8144 unused inodes
  Free blocks: 65536-98303
  Free inodes: 16289-24432
```

الآن سوف نقفز إلى أول مجموعة كتل في مجموعة الكتل المرنة التالية Flexible Block Group 2 وستكون Block Group 16 :

- هذه أول مجموعة كتل لكن لا تتضمن backup Superblock، و Group Descriptors، و Reserved GDT blocks. (بسبب تمكين sparse_super)
- وتتضمن Data Block Bitmaps، و inode Bitmaps، و inode Tables لجميع مجموعات الكتل الأخرى (16-31) في مجموعة الكتل المرنة هذه.

```
Group 16: (Blocks 524288-557055) [INODE_UNINIT, ITABLE_ZEROED]
  Checksum 0x8ab4, unused inodes 8144
  Block bitmap at 524288 (+0), Inode bitmap at 524304 (+16)
  Inode table at 524320-524828 (+32)
  24592 free blocks, 8144 free inodes, 0 directories, 8144 unused inodes
  Free blocks: 532464-557055
  Free inodes: 130305-138448
```

ملاحظة: الأعلام التي تظهر في مجموعة الكتل: هي لإبطال تهيئة initializing / تصفير zeroing بعض الهياكل. ولتقليل من وقت عمل أداة mkfs (راجع فقرة lazy block group initialization)

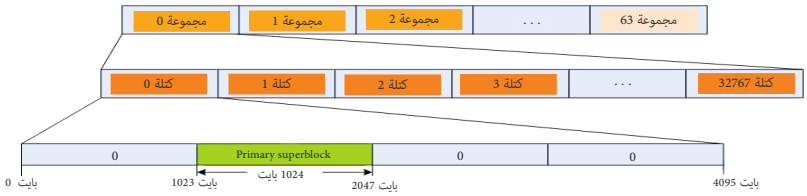
56. ^ عدد inodes الإجمالي في نظام الملفات. هذه القيمة يجب أن تكون أقل أو تساوي (s_inodes_per_group * number of block groups) وأن تعادل حاصل inodes المحدد في كل مجموعة كتل.
57. ^ عدد الكتل الإجمالي في نظام الملفات هذه القيمة يجب أن تكون أقل أو تساوي (s_blocks_per_group * number of block groups) وأن تعادل حاصل الكتل المحدد في كل مجموعة كتل.

58. ^ 32 بت المنخفضة (من العدد الصحيح 64 بت). تشير إلى عدد الكتل الإجمالي المحجوزة من قبل المستخدم الجذر. هذا مفيد جدا في حالة قام أحد المستخدمين (بقصد أو بغير قصد)، بشغل نظام الملفات إلى سعته القصوى، حينذاك هذه الكتل الحرة ستكون تحت تصرف المستخدم الجذر، كي يستطيع تحرير وحفظ ملفات التضبيط / الإعدادات. وبدون هذه الميزة، لا يستطيع المستخدم الجذر الولوج إلى نظام الملفات، تقنيا، هذه تستخدمها العمليات ذات الامتياز privileged processes وتسمح باستمرار عمل ما يسمى عفريت النظام مثل syslogd، هذه الميزة أيضا تسمح بوجود "جيوب فارغة" بين الملفات تساعد في منع التجزئة عند تعديل الملفات. هذه الميزة تصبح بلا أهمية إذا كان استخدام نظام الملفات (وحدة التخزين) من أجل فقط الأرشيف الطويل الأمد (حيث لا تتغير الملفات كثيرا). القيمة الابتدائية دائما 5%.(The default percentage is 5%.) ويمكن تغييرها بخيار 0 m- مع أداة tune2fs. (لكن لا ينصح بذلك في الأنظمة الصغرى) راجع استخدام MKE2FS / TUNE2FS

59. ^ 32 بت تشير إلى العدد الإجمالي للكتل الحرة، والتي تشمل عدد الكتل المحجوزة (s_r_blocks_count). هذا سيكون حاصل جميع الكتل blocks الحرة في جميع مجموعات الكتل.

60. ^ 32 بت تشير إلى العدد الإجمالي للمؤشرات الفهرسة الحرة. هذا سيكون حاصل جميع inodes في جميع مجموعات الكتل.

61. ^ 32 بت تعرف بأول كتلة بيانات، أي هوية الكتلة التي تتضمن superblock. لاحظ أن هذه القيمة دائما 0 في الأنظمة التي تستخدم حجم كتلة أكبر من 1k، ودائما 1 في الأنظمة التي تستخدم حجم كتلة 1k . وكتلة superblock دائما تقع عند إزاحة بايت 1024 من بداية الملف أو بداية جهاز الكتل أو بداية القسم (الذي عادة يكون أول بايت من القطاع الثالث). بنية superblock باستثناء بعض التغييرات هي تقريبا نفسها في جميع أنظمة ext2/3/4. في الأنظمة التي تستخدم حجم الكتلة 1 كيلوبايت، الكتلة 0 ليست جزء من مجموعة الكتل 0 block group. لأن هذه الأخيرة دائما تبدأ مع كتلة superblock. لهذا، عند استخدام حجم الكتلة 1k، مجموعة الكتل 0 تبدأ في الكتلة 1. وعند استخدام حجم الكتلة الأكبر، (كما تظهر في الخطاطة التالية) تبدأ مع الكتلة 0. (راجع s_first_data_block في superblock).



مثال : في هذه الخطاطة يظهر موقع الكتلة العليا super block على نظام ملفات يملك 63 مجموعة (هنا حجم الكتلة كان 4096 بايت) كل مجموعة بحجم 32768 كتلة باستثناء المجموعة الأخيرة

- في حالة كان حجم الكتلة = 1024 بايت، في المجموعة 0، توسع الكتلة 0 بالمستخدمة ولا تستخدم. والكتلة 1 تتضمن Primary superblock، المجموعة 0 تتضمن أيضا كتل البيانات الوصفية: Group descriptors و Reserved GDT blocks و Block bitmap و Inode bitmap، و Inode table، وكتل البيانات Data Blocks.
- في حالة كان حجم الكتلة أكبر من 1024 بايت (كما في الخطاطة أعلاه، 4096 بايت)، في المجموعة 0، يتم حشو أول 1024 بايت من الكتلة 0 ولا تستخدم. والكتلة 0 تتضمن Primary superblock (بعد إزاحة 1023)، المجموعة 0 تتضمن أيضا كتل البيانات الوصفية: Group descriptors و Reserved GDT blocks و Block bitmap و Inode bitmap، و Inode table، وكتل البيانات Data Blocks.
  - عند إنشاء نظام الملفات مثلا بهذه الطريقة: mkfs -t ext4 -b 1024 /dev/sda1. النظام يستخدم كتل 1 كيلوبايت، وعند فحص الكتلة الأولى : dumpe2fs -h /dev/sda1 | grep "First block" سوف تلاحظ أن الكتلة 0 مهملة بالكامل وخرج أداة dumpe2fs يشير إلى الرقم واحد : "First block: 1" يبدوا أن هذه القيمة إما أن تكون 1 أو تكون 0 في معظم الوقت.
- بعض مجموعات الكتل (1، 3، 5، 7...إلى آخره) تتضمن نسخ من superblock و Group descriptors. وكتل Reserved GDT blocks و Block bitmap و Inode bitmap، و Inode table، و Data Blocks موقع تلك المصفوفات الثنائية bitmaps وجدول Inode table يشير إليه واصف المجموعة الخاص Group descriptor، ولذلك الترتيب قد يختلف عن هذا.
- في حال تمكين الميزتين sparse_super + flex_bg، أغلب مجموعات الكتل سوف تتضمن فقط كتل البيانات (راجع flex_bg).

62. ^ أ، ب. حجم الكتلة block size يجب أن لا يقل عن 1024 بايت، ولا يزيد عن حجم الصفحة المدعوم في البنية (المعمارية)، مثلا في x86. حجم الكتلة الصالح سيكون 1 كيلوبايت، 2 كيلوبايت، 4 كيلوبايت

63. ^ 32 بت تشير إلى العدد الإجمالي للكتل في كل مجموعة. الجمع بين هذه القيمة و s_first_data_block يمكن استخدامه في رسم حدود مجموعات الكتل.

64. ^ 32-بت تشير إلى إجمالي inodes في كل مجموعة. القيمة أيضا تستخدم في تحديد حجم inode bitmap في كل مجموعة كتل. لاحظ أنه لا يمكن أن يكون هناك أكثر من (حجم الكتلة بالبايت × 8) عدد inodes لكل مجموعة (مثال 4096 × 8 = 32768) لأن inode bitmap يجب أن تتناسب داخل كتلة واحدة. هذه القيمة يجب أن تكون ضرب عدد inodes التي يمكن أن تتناسب في الكتلة.

65. ^ 16 بت تشير إلى وضعية نظام الملفات. إذا كان نظام الملفات موصول القيمة ستكون 0x0002. بعد فصل نظام الملفات على نحو نظيف، هذه القيمة تتحول إلى 0x0001. عند وصل نظام الملفات، إذا صادف وجود قيمة 0x0002 صالحة هذا يعني أن نظام الملفات ليس مفصول على نحو نظيف مع احتمال وجود أخطاء تحتاج إلى إصلاح. في لينكس هذا يعني عمل fsck.

66. ^ 16 بت تشير إلى ما ينبغي على مشغل نظام الملفات (نظام التشغيل) عمله عندما يصادف خطأ في نظام الملفات. هذه القيم يمكن ضبطها عند إنشاء نظام الملفات، وستكون إحدى ثلاثة قيم.

67. ^ 32 بت تستخدم للإشارة إلى أول inode صالح للاستخدام مع الملفات المعيارية. في المراجعة 0، أول inode غير محجوز كان 11. في المراجعة 1 واللاحقة هذه القيمة يمكن تعيينها إلى أية قيمة.

68. ^ 16 بت تشير إلى حجم بنية inode بعدد بايتات، في المراجعة 0، هذه القيمة كانت دائما 128 بايت. وفي المراجعة 1 واللاحقة، هذه القيمة يجب أن تكون بالضبط قوة العدد 2 و يجب أن تكون أصغر أو تساوي حجم الكتلة.

69. ^ 16 بت تستخدم للإشارة إلى رقم مجموعة الكتل التي تستضيف بنية الكتلة العليا superblock. (البيانات الوصفية العامة للنظام الملفات) يمكن استخدام هذا في إعادة بناء نظام الملفات من أية نسخة superblock.

70. ^ 32 بت (قناع بت) أعلام الميزّات المتوافقة. تطبق نظام الملفات سيكون حر في دعم أو عدم دعم هذه الميزّات، دون خطر إتلاف البيانات الوصفية. والنواة ما زال بإمكانها القراءة / الكتابة إلى نظام الملفات حتى وإن لم تفهم العلم، لكن أداة fsck لا يجب أن تفعل ذلك.

71. ^ 32 بت (قناع بت) أعلام الميزّات الغير متوافقة. تطبق نظام الملفات يجب أن يرفض وصل نظام الملفات إذا كانت الميزة المحددة بدون دعم. التطبيق الذي لا يدعم هذه الميزّات سيكون غير قادر على استخدام نظام الملفات بالشكل الصحيح. مثلا، إن كان ضغط البيانات مستخدم، بعد قراءة ملف تنفذي من القرص، سيكون غير صالح للاستعمال إن كان النظام لا يعرف فك ضغط الملف. والنواة أو أداة fsck يجب أن تتوقف إذا لم تفهم إحدى هذه بتات.

72. ^ 32 بت (قناع بت) أعلام الميزّات المتوافقة – في وضعية القراءة فقط، تطبق نظام الملفات ينبغي أن يوصل في وضعية القراءة فقط، إذا كانت الميزة المحددة بدون دعم. والنواة ما زال بإمكانها وصل نظام الملفات في وضعية القراءة فقط إذا لم تفهم إحدى هذه بتات.

73. ^ قيمة 32 بت تشير إلى أول inode في لائحة من inodes من أجل الحذف.

74. ^ مصفوفة من 4 قيم 32-بت تتضمن القيمة الابتدائية المستخدمة من قبل خوارزمية الهاش في فهرسة الأدلة

75. ^ 32 بت (تحمل إشارة) تشير إلى حجم الملف بالبايت. في المراجعة 0 وفي المراجعات اللاحقة، تمثل 32 بت المنخفضة من حجم الملف من أجل الملفات الاعتيادية فقط؛ 32 بت العليا تقع في i_dir_acl

76. ^ أختام زمنية في inode بعدد الثواني (توقيت يونكس)
    - حقل i_atime بقيمة 32 بت تشير إلى زمن النفاذ آخر مرة إلى الملف Last access time،
      - لكن إذا تم تعيين علم EA_INODE، هذا inode يخزن قيمة الخاصية الممتدة والحقل (i_atime) يحوي تدقيق مجموع القيمة.
    - حقل i_ctime بقيمة 32 بت تشير إلى زمن تغيير آخر مرة مؤشر الفهرسة (نفسه) Last inode change time (أي تغيير البيانات الوصفية)
      - لكن إذا تم تعيين علم EA_INODE، هذا inode يخزن قيمة الخاصية الممتدة والحقل (i_ctime) يحوي 32 بت المنخفضة من التعداد المرجعي للقيمة الخاصة.
    - حقل I_mtime بقيمة 32 بت تشير إلى زمن تعديل البيانات آخر مرة Last data modification time (تعديل inode)
      - لكن إذا تم تعيين علم EA_INODE، هذا inode يخزن قيمة الخاصية الممتدة والحقل (I_mtime) يحوي رقم inode الذي يمتلك الخاصية الممتدة.

77. ^ أ، ب، 16 بت تشير إلى تَعْداد الروابط الصلبة، معظم الملفات تملك رابط واحد.عادة، EXT4 لا يسمح بامتلاك inode أكثر من 64,000 رابط صلب. هذا يطبق على الملفات والأدلة، ويعني أن الدليل لن يضم أكثر من 64,998 دليل ثانوي (كل مدخلة '..' في الدليل الفرعي تعد رابط صلب، تماما مثل مدخلة '.' في الدليل نفسه)في حالة تمكين ميزة DIR_NLINK، نظام الملفات سيدعم أكثر من 64,998 دليل فرعي بتعيين القيمة 1 في هذا الحقل للإشارة إلى أن عدد الروابط الصلبة مجهول. وصلات الرمزية لا تأثر على تعداد الروابط لكن عندما يصل التعداد إلى الصفر inode وجميع كتله تصبح حرة.

78. ^ 32 بت تشير إلى تَعْداد"block"هذا الحقل في ext2/3 يسمى "Sector count" أو "Blocks count ”(عدد الكتل المحجوزة للملف/inode، الذي يمثل بوحدات من الكتل المنطقية أو قطاع 512 بايت) (مثلا 8 إذن 8 * 512 = 4096B)
   - في حال عدم تعيين علم ميزة huge_file، (في superblock) على نظام الملفات، الملف يستخدم أو يستهلك كتل 512-بايت i_blocks_lo على القرص.
   - في حال تعيين huge_file وعدم تعيين EXT4_HUGE_FILE_FL في inode.i_flags (في inode) الملف يستهلك كتل 512-بايت (i_blocks_hi << 32) + i_blocks_lo على القرص.
   - في حال تعيين huge_file مع تعيين EXT4_HUGE_FILE_FL IS، هذا الملف يستهلك كتل نظام الملفات (i_blocks_hi << 32) + i_blocks_lo على القرص.

79. ^ رغم أن EXT4_JOURNAL_DATA_FL و EXT4_EXTENTS_FL يمكن تعيينها بأداة setattr، لكنها ليست في قناع النواة EXT4_FL_USER_MODIFIABLE، لأنها تحتاج إلى معالجة إعدادات هذه الأعلام بأسلوب خاص وهي مقنعة في تشكيلة الأعلام المحفوظة مباشرة إلى i_flags

80. ^ 32 بت تشير إلى إصدارة inode. لكن، عند تعيين علم EA_INODE، هذا inode يخزن قيمة الخاصية الممتدة والحقل هذا (l_i_version) يتضمن 32 بت العليا من التعداد المرجعي للقيمة الخاصية.

81. ^ نظريا، يمكن تعيين هذه القيمة للإشارة إلى الكتلة التي تتضمن الخصائص الممتدة للدليل أو الملف الخاص. في المراجعة 0 هذه 32 بت دائما 0، وفي المراجعة 1، هذه 32 بت تتضمن 32 بت العليا من حجم ملف 64 بت من أجل الملفات الاعتيادية. لينكس يعين هذه القيمة إلى 0 إن كان الملف ليس ملف اعتيادي (أي، ملف جهاز كتل، ملف دليل،... إلى آخره).

82. ^ 32 بت هوية (مستخدم) مؤلف الملف المخصص assigned file author ! (بعض وثائق ext2 تقول: إذا تم تعيين هذه القيمة إلى -1 يستخدم معرف المستخدم POSIX)

83. ^ 8 بت تشير إلى عدد المستويات الغير مباشرة الموجودة في هذا الهاش.

84. ^ الأجراء هو عملية ذرية ؛ رسالة أو تعديل للبيانات أو إجراء آخر يضمن إما تنفيذه كاملا، أو لا يتم تنفيذه إطلاقا (مثل، إجراء قاعدة السانات)

85. ^ قناع (حجاب) mask هو نمط أو متسلسلة من بتات تستخدم في عمليات بتّي؛ قناع البت Bit mask (المعالجة عن طريق أصغر جزء من المعلومة) و mask قد يعني :
   - تعيين set أو إلغاء تعيين unset (بتات محددة، أو أرقام ثنائية داخل القيمة) بواسطة قناع البت Bit mask
   - تعطيل (مقاطعة، إلى آخره) عن طريق إلغاء تعيين البت المقترن.

86. ^ أ، ب، ت، ث، ج، ح، بما أن Superblock تتضمن معلومات مهمة جدا، خسارتها تعني فشل كامل نظام الملفات (أي لا يمكن قراءة أو إقلاع نظام الملفات بدون المعاملات في الجدول أعلاه)، ستكون هناك دائما نسخ احتياطية من Superblock إلى جانب Group Descriptors في بعض أو جميع مجموعات الكتل ! في أنظمة الملفات الصغرى، كما في هذا المثال الذي يملك 120 مجموعة، المستخدم لا يحتاج إلى 120 نسخة احتياطية !؟. لهذا السبب، تستخدم ميزة sparse_super التي تسمح بوجود نسخ مكررة من Superblock و Group Descriptors في بعض المجموعات فقط. تحديدا، في مجموعة الكتل 0 (التي تتضمن النسخة الأولية) والمجموعات أس العدد 1 ،3 ،5، أو 7، إلى آخره. مثلا، مجموعات الكتل في المثال تمتد من 0 إلى 119 (أي 120 ضمنيا)، إذن النسخ المكررة ستكون في: 1 3 5 7 9 25 27 49 81. إذا ضربنا كل رقم بالعدد 32768 (عدد الكتل في المجموعة) نحصل على أرقام الكتل التالية (تظهر في خرج mkfs)

1	3	5	7	9	25	27	49	81
32768	98304	163840	229376	294912	819200	884736	1605632	2654208

تمكين ميزة **sparse_super2** يسمح بوجود نسختين فقط من superblocks و block group descriptors. حقل s_backup_bgs[2] في superblock سيشير إلى المجموعتين حيث توجد النسختين. التي عادة تكون إحداها في بداية المجموعة #1 block group، والأخرى في مجموعة الكتل الأخيرة في نظام الملفات. هذه الميزة تسمح بزيادة نسبة كتل البيانات المتماسة على القرص!

87. ^ حقول تخزين تدقيق المجامع الجارية *h_chksum في ترويسة الكتلة التنفيذ Commit Block :
   في حال تعيين FEATURE_COMPAT_CHECKSUM (checksum v1)، حقول *h_chksum تستخدم في تخزين تدقيق مجموع كتل التوصيف descriptor والبيانات data.
   في حال تعيين FEATURE_INCOMPAT_CSUM_V2 (checksum v2)، حقل h_chksum يستخدم في تخزين crc32c(uuid+commit_block). كل كتلة بيانات وصفية في journal تحصل على تدقيق مجموع خاص، وتدقيق مجاميع كتل البيانات data يخزن في لاحقة كتلة قيد الحوادث journal_block_tag (في descriptor). حقول *h_chksum الأخرى لا تستخدم.
   في حال تعيين FEATURE_INCOMPAT_CSUM_V3، كتلة التوصيف descriptor تستخدم journal_block_tag3_t لتخزين كامل تدقيق مجموع 32-بت أي شيء ما عدا ذلك سيكون مثل v2.
   تدقيق مجموع v1 و v2 و v3 هي ميزات متنافية (أي، لا يمكن أن تقع في نفس الوقت) [هذه المعلومات كانت من ملف: kernel-jbd.h]

88. ^ الميزة METADATA_CSUM تمكن أيضا تدقيق مجاميع توصيف المجموعات GDT_CSUM. في حالة تعيين METADATA_CSUM، تدقيق مجاميع توصيف المجموعات يستخدم نفس الخوارزمية مثل تدقيق مجاميع هياكل البيانات الأخرى. رغم ذلك، بتات الميزتان METADATA_CSUM و GDT_CSUM متنافيتان (أي، لا يمكن أن تقع في نفس الوقت)

89. ^ حقل يشير إلى مساحة من 32 بايت من أجل تخزين تدقيق المجامع.
   في حالة تعيين INCOMPAT_CSUM_V2 أو INCOMPAT_CSUM_V3، أول __be32 ستكون تدقيق مجموع journal UUID وكامل commit block، وهذا الحقل سيكون صفر.
   في حالة تعيين COMPAT_CHECKSUM، أول __be32 ستكون crc32 لجميع الكتل التي تم كتبتها بالفعل إلى الإجراء transaction.

90. ^ 16 بت تشير إلى # عدد ثواني انتظار فحص MMP. نظريا، هذه آلية لتسجيل المضيف والجهاز الذي وصل نظام الملفات، كي تمنع الوصل المتعدد لكن الميزة تبدوا غير مطبقة.

91. ^ 16 بت تشير إلى تَعْداد inodes الغير مستخدمة، في حالة التعيين، لا تحتاج فحص ما بعد المدخلة (sb.s_inodes_per_group - gdt.bg_itable_unused) في جدول inode table لهذه المجموعة.

92. ^ 16 بت تشير إلى تدقيق مجموع توصيف المجموعة Group descriptor checksum :
   في حالة تعيين ميزة RO_COMPAT_GDT_CSUM، سيكون crc16 (sb_uuid+group+desc)
   في حالة تعيين ميزة RO_COMPAT_METADATA_CSUM سيكون crc32c (sb_uuid+group_desc) و 0xFFFF

93. ^ في النسخة v2 كل كتلة بيانات وصفية في journal تحصل على تدقيق مجموع خاص. ولواحق الكتل block tags في جدول التوصيف تتضمن تدقيق مجاميع لكل كتلة بيانات في journal.

94. ^ النسخة v3 مثل V2 لكن حجم لاحقة كتلة قيد الحوادث journal block tag ثابت بغض النظر عن حجم أعداد الكتل.

95. ^ أ، ب، ت، ث، ج، ح، ميزة meta_bg تسمح بإعادة تحجيم resized نظام الملفات في وضع المتصل on-line دون الحاجة إلى حجز مساحة للنمو block group descriptors. هذا المخطط يستخدم أيضا في إعادة تحجيم نظم الملفات الأكبر من 2^32 كتلة. ولا يوصى بتعيين هذه الميزة في زمن إنشاء نظام الملفات، لأن هذا النظام البديل لتخزين block group descriptors سيبطئ من وصل نظام الملفات، وألأنويه الأحدث يمكنها آليا تعيين هذه الميزة عند الحاجة لها عند عمل إعادة تحجيم في وضع المتصل online resize ولا تتوفر مساحة محجوزة في inode resize (راجع الملاحظة رقم 50)

96. ^ إعادة تحجيم متصل Online Resize تعني إمكانية تغيير حجم الأقراص التي هي قيد العمل دون الحاجة لتوقف أو إعادة التشغيل أو اقتطاع مساحة من قرص موجود مسبقًا وإنشاء واحد جديد،

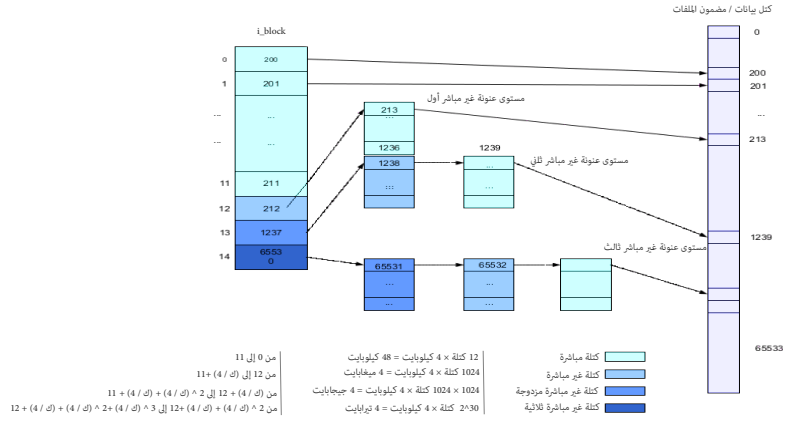
97. ^ أ، ب، ت، عنونة الكتل المباشرة والغير مباشرة تملك 12 رابط مباشر، ورابط غير مباشر، ورابط غير مباشر مزدوج، ورابط غير مباشر ثلاثي (المجموع 15، لأن 60 بايت ÷ 4 = 15) هنا كلمة غير مباشر (ة) Indirect تعني أن الكتلة في العنوان ليست كتلة بيانات بل كتلة مستوى آخر تعرض عناوين كتل أخرى، وقد تملك روابط غير مباشرة من 1 واحد، 2 واحد (مزدوج)، 3 واحد (ثلاثي).
   باعتبار أن حجم مؤشر الكتلة الغير مباشر(ة) 4 بايت، إذن كتلة 4 كيلوبايت يمكنها الإشارة إلى 1024 كتلة تتضمن بيانات لأن 4 بايت × 1024 = 4 كيلوبايت، بهذا يمكن عنونة 11+1024+1024*1024+1024*1024*1024 كتلة. لكن، لا يمكن أن يكون العنوان 64-بت (لأن كل عنوان 4 بايت).

نظام العنونة هذا استخدم في ext 2 و ext3، والآن لا يستخدم في ext4. الذي يستخدم ربط مديات extents. (أنظر للملاحظة التالية رقم 100)

حقل i_block هو مصفوفة تتضمن عناوين للكتل البيانات المرتبطة بـ inode. مبدئيا هناك 15 مدخلة، عنونة الكتل الغير مباشرة Indirection تبدأ من المباشرة وتنتهي بالغير مباشرة الثلاثية.

مثلا، لحساب عنونة كتل البيانات، نفترض أن حجم كتلة 4 كيلوبايت، أي 4096 = " ك"، في هذه الصيغ القسمة ستكون على 4 لأن كل رقم كتلة منطقية مخزن في 4 بايت. مع حجز 60 بايت في inode :

- المدخلات الاثنا عشر الأولى في مصفوفة []i_block تتضمن عناوين إلى أرقام الكتل المنطقية من 0 إلى 11 (أي 12 مؤشر تعمل مباشرة من داخل inode)
- المدخلة الغير مباشرة الفذة عند المؤشر 12 توصل في النهاية إلى كتل تبدأ بالكتلة 12 حتى الكتلة (ك÷4)+11 ؛ إذن، الكتلة الغير مباشرة الفذة لعنونة الكتل ستكون: من 12 إلى 1035
- المدخلة الغير مباشرة المزدوجة عند المؤشر 13 لعنونة الكتل من (ك÷4)+12 إلى (ك÷4)^2 + (ك÷4)+11 ؛ إذن، ستكون الكتل من 1036 إلى 1049611
- المدخلة الغير مباشرة الثلاثية عند المؤشر 14 لعنونة الكتل من (ك÷4)^2 + (ك÷4)+12 إلى (ك÷4)^3 + (ك÷4)^2 + (ك÷4)+12 هذا يعني كتل كثيرة: من 1049612 إلى 1074791435 (نظريا)

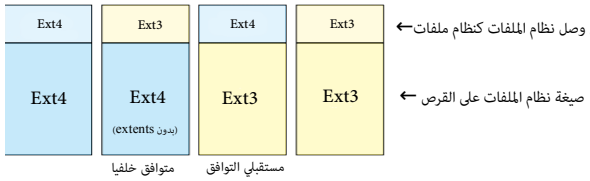


مخطط ربط الكتل (15 × "32-بت") أو العنونة الغير مباشرة كان يستخدم في ext2/3 والآن هذا المخطط يستخدم فقط للتوافق خلفيا ؛ عند التحويل بين ext3 و ext4

- طريقة إعداد عنونة الكتل هذه تناسب الملفات الصغرى. إذا كان الملف يملك 12 كتلة أو أقل، ستحتاج فقط النظر في مصفوفة i_block للوصول مباشرة إلى كتلة البيانات المطلوبة. الملفات الطويلة، ربما تحتاج إلى نفاذ متعدد للقرص (أي سيكون هناك الكثير من السعي عند البحث المعمق عن الملف)، اعتمادا على مستوى العنونة الغير مباشرة Indirection.
- حجم الكتلة، طبعا له تأثير مباشر على كمية العنونة الغير مباشرة Indirection. مع أن العنونة الغير مباشرة الثلاثية triple-indirection قد تكون ضرورية للملف الكبير جدا على نظام الملفات مع كتل بحجم 1k، مع حجم الكتلة 4k، أي ملف من 2 جيجابايت أو أقل (أقصى حجم ملف مسموح به في ext2 و ext3 على البنية المعمارية 32 بت) يمكن عنونته في الغالب بالعنونة الغير مباشرة المزدوجة double indirection.

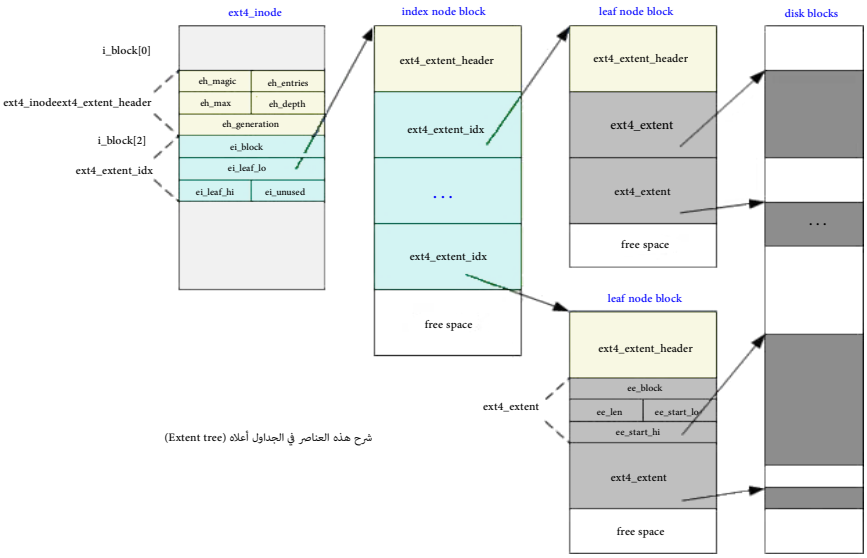
98. ^ أ ب ت ث ج بطانة، حشو، حشوة padding تعني محارف إضافية مثل الفراغات تضاف إلى نهاية تسجيلة حتى تصبح بطول ثابت. عادة هذا يستخدم في محاذاة هياكل البيانات وفي علم التشفير.

99. ^ أ ب ت ث التوافقية المستقبلية والتوافقية الرجعية! (التوافق خلفيا)



100. ^ أ ب ت مخطط شجرة المديات Extent Tree

- شجرة بحث ثنائية متزنة ذاتيا Balanced tree في شكل متتابعات (تتابعات) من offset:block:length (طول : عنوان كتلة القرص : إزاحة كتلة الملف)
- مدخلة واحدة لكل رتل / تنفيذ متتالي consecutive run
- المديات extents الأربعة الأولى ستكون داخل inode
- هذه بنية شجرية متراصة (أو مدمجة) تحمل بسرعة
- ترتبط بعمل الميزتين Multiple block allocation و Delayed allocation (ملاحظة: تطبيق كلا الميزتين لا يغير أو يأثر على تهيئة القرص)
- مفيدة جدا في التخصيص المسبق للمساحة.preallocating space



101. ^ 16 بت تشير إلى عمق عقدة المدى هذه في شجرة المديات. القيمة 0 تعني أن extent node هذه تشير إلى كتل بيانات ؛ ما عدا ذلك، عقدة المدى هذه تشير إلى عقد مدى أخرى. شجرة المديات يمكن أن تكون على الأكثر بعمق 5 ؛ رقم الكتلة المنطقية يمكن أن يكون على الأكثر 2^32، وأصغر رقم n يمكن أن يفي بشرط التالي، سيكون خمسة 5: 4*(n^(12/(blocksize - 12))) =< 2^32

102. ^ 16 بت تشير إلى عدد الكتل التي يغطيها هذا المدى. إذا كانت قيمة هذا الحقل أقل من أو يساوي 32768 يتم تهيئة المدى. وإذا كانت قيمة هذا الحقل أكبر من 32768 لا يتم تهيئة المدى وطول المدى الفعلي سيكون ee_len – 32768.

ومن ثم الطول الأقصى للمدى المهيئ (initialized) هو 32768 كتلة (2^15)، والطول الأقصى للمدى الغير مهيئ (لكن preallocated) هو 32767. (1 – 2^15)

توضيح: قيمة حقل حجم المدى 16-بت فقط. منها بت العليا high bit محجوز لوسم (أو تعليم) المدى بالمدى المخصص مسبقا preallocated extent، لذلك كتل المدى ستكون 32 كيلوبايت فقط. على افتراض أن حجم الكتلة هو 4 كيلوبايت، هذا يعني أن كل مدى سيملك فقط 128 ميغابايت من البيانات. وبالتالي، ملف (كما في المثال التالي) من 4 جيجابايت سيتطلب على الأقل 32 مدى، وحتى ولو افترضنا وجود 32 رتل متماس (متجاور) من كتل 32 كيلوبايت. غالبا ستكون المديات أكثر من 32 مدى، بعضها لا يستخدم كامل 128 ميغابايت.

في هذا الاختبار، بعد إنشاء ملف بحجم 4 جيجابايت، نستخدم نفس التقنية (كما في الأمثلة الأخرى أدناه) في فك بنية شجرة المديات وإيجاد كتلة البيانات التي تتضمن المديات الفعلية للملف:

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 0A F3 34 00 54 01 00 00 00 00 00 00 00 00 00 00 |..4.T...........|
0010 00 30 00 00 00 50 A4 01 00 30 00 00 00 80 00 00 |.0...P...0......|
0020 00 A0 A4 01 00 B0 00 00 00 80 00 00 00 20 A5 01 |................|
0030 00 30 01 00 00 58 00 00 00 A0 A5 01 00 88 01 00 |.0...X..........|
0040 00 08 00 00 00 30 A6 01 00 90 01 00 00 08 00 00 |.....0..........|
0050 00 40 A6 01 00 98 01 00 00 10 00 00 00 20 A6 01 |.@.............|
0060 A8 01 00 00 00 50 00 00 00 50 A6 01 00 F8 01 00 |.....P...P......|
0070 00 18 00 00 00 A8 A6 01 00 10 02 00 00 28 00 00 |.............(..|
0080 00 C8 A6 01 00 38 02 00 00 08 00 00 00 F8 A6 01 |.....8..........|
0090 00 40 02 00 00 80 00 00 00 48 A7 01 00 C0 02 00 |.@.......H......|
00A0 00 30 00 00 00 D0 A7 01 00 F0 02 00 00 80 00 00 |.0..............|
00B0 00 88 A8 01 00 70 03 00 00 28 00 00 00 08 A9 01 |.....P...(......|
00C0 00 98 03 00 00 80 00 00 00 70 A9 01 00 18 04 00 |.........P......|
00D0 00 10 00 00 00 F0 A9 01 00 28 04 00 00 10 00 00 |.........(......|
00E0 00 F8 AA 01 00 38 04 00 00 08 00 00 00 30 AB 01 |.....8.......0..|
00F0 00 40 04 00 00 70 00 00 00 48 AB 01 00 B0 04 00 |.@...P...H...0..|
```

لاحظ أن عدد مديات الملف في ترويسة المديات كان 52 (0x0034) مدى. لكن ما يثير الانتباه, هو بنية المدى الثانية. هنا المدى يبدأ عند الإزاحة المنطقية 0x00003000 (الكتلة 12288 من بداية الملف) والكتلة الفيزيائية 0x0000 01A4A000 (رقم الكتلة 27566080). المفاجأة كانت هنا في حجم المدى 0x8000. الذي في الثنائي 16 بت تساوي 1000000000000000، مع تعيين بت العليا، بينما 15 بت المنخفضة تكون جميعا أصفار. السبب هو أن ext4 يستخدم بت العليا في وسم المدى بالمدى المخصص مسبقا preallocated extent، هذا يعني مدى مخصص مسبقا بقيم الصفر. إذن ما الذي يجري هنا ؟

أولا، نقتبس من نص محادثة جرت بين بعض مطوري Ext4 (+ الشفرة والتعليقات) ما يلي:

```
Amit will first be merging in Andreas' patch to fallocate, which allows initialized extents to be the full 32768 blocks.
Uninitialized extents are limited to 32767 blocks. Amit will also add comments to this, and have the update patches ready by tomorrow.
#define EXT_MAX_LEN ((1UL << 15) - 1)
EXT_INIT_MAX_LEN is the maximum number of blocks we can have in an initialized extent. This is 2^15 and not (2^16 - 1), since we use the MSB of
ee_len field in the extent data structure to signify if this particular extent is an initialized extent or an uninitialized (i.e. preallocated).
EXT_UNINIT_MAX_LEN is the maximum number of blocks we can have in an uninitialized extent. If ee_len is <= 0x8000, it is an initialized extent.
Otherwise, it is an uninitialized one. In other words, if MSB of ee_len is set, it is an uninitialized extent with only one special scenario when
ee_len = 0x8000.
In this case we can not have an uninitialized extent of zero length and thus we make it as a special case of initialized extent with 0x8000
length. This way we get better extent-to-group alignment for initialized extents. Hence, the maximum number of blocks we can have in an
*initialized* extent is 2^15 (32768) and in an *uninitialized* extent is 2^15-1 (32767).
#define EXT_INIT_MAX_LEN (1UL << 15)
#define EXT_UNINIT_MAX_LEN (EXT_INIT_MAX_LEN - 1)
```

إذا كان كل بت في المصفوفة الثنائية للكتل block bitmap يتعقب حالة استخدام كتلة واحدة في مجموعة الكتل، إذن، الكتل التي يمكن تعقبها باستخدام كتلة block bitmap سيكون 8x(block size) أو 32 كيلوبايت في أنظمة الملفات التي تستخدم حجم الكتلة 4 كيلوبايت. و block bitmap إلى جانب الكتل المحجوزة للمؤشرات الفهرسة inodes و inode bitmap (عادة EXT يخصص inode واحد لكل 4 كتل في مجموعة الكتل) ونسخ superblock والبيانات الوصفية للنظام الملفات الأخرى عادة تخزن مباشرة قبل كتل البيانات في مجموعة الكتل. جميع هذه البيانات الوصفية تعني أنك لن تجد أكثر من 32K من كتل البيانات المتجاورة في نظام ملفات EXT - هذا فقط إذا كانت مجموعة الكتل المعنية حاليا غير مستخدمة.

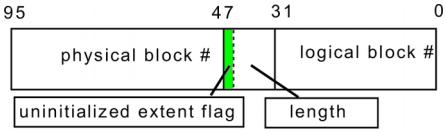
والآن دعنا نفكر في ذلك في سياق حقل extent size في ext4. هذا الحقل بقيمة 16-بت، لكن بت العليا محجوز، هذا يعني أن extent يمكن أن يتضمن فقط ما يصل إلى 2^{15}-1 كتلة (أو 32767 كتلة) أي أقل بكتلة واحدة من عدد الكتل في مجموعة الكتل الواحدة. وهذا يعتبر تبذير في المساحة.

لكن لماذا البت العليا في extent size محجوز؟. كي يستطيع نظام الملفات وسم مديات معينة بالغير مهيئة لكنها محجوزة "uninitialized but reserved". استراتيجية التخصيص المسبق هذه "preallocation" تسمح للنظام ملفات ext4 منع الملفات الأخرى من استخدام كتل معينة، إذا ظن النظام أن الملف سيحتاج هذه الكتل مستقبلا، ومن ثم تجنب تجزئة الملف الذي سينمو.

للسماح للمديات بشغل كامل مجموعة الكتل، مطوري ext4 استخدموا حيلة حجم المدى 0x8000 التي تعني مدى بدون تهيئة يتضمن كتل صفرية an uninitialized extent with zero blocks in it. لكن لماذا نخصص مسبقا كتل صفرية ؟. لذلك، مطوري ext4 أضافوا حالة خاصة تقول أن القيمة 0x8000 تعني مدى مخصص allocated extent من كتل 32k كاملة في مجموعة الكتل.

جميع القيم الأخرى التي تملك تعيين بت العليا high bit تعني مدى مخصص مسبقا لكن بدون تهيئة a preallocated but uninitialized extent وطول المدى تحدده 15 بت الأخرى في extent size. لكن ذلك يعني إمكانية فقط تخصيص مسبقا ما يصل إلى 1 – 2^15 كتلة، أو 32767 كتلة، أي أقل بكتلة واحدة من العدد الأقصى للكتل في مجموعة الكتل. وهذا بالضبط (ما يحاول اخبارنا به) أو ما جاء في نص الشفرة والتعليق أعلاه، إذن الجواب باختصار، حجم المدى extent size بقيمة 0x8000 تعني مدى مخصص allocated extent بطول كتل 32k. وأية قيمة أصغر ستكون أيضا مدى مخصص، لأن بت العليا لن يعين. وأية قيمة أكبر من 0x8000 ستكون مدى مخصص مسبقا preallocated extent تحدد طوله 15 بت المخفضة في القيمة.

ext4_extent structure



راجع أعلاه جداول extent tree

ext4_extent_header	ext4_extent_idx	ext4_extent	ext4_extent_tail
eh_magic eh_entries eh_max eh_depth eh_generation	ei_block ei_leaf_lo ei_leaf_hi ei_unused	ee_block ee_len ee_start_hi ee_start_lo	eb_checksum

- يستخدم بت MSB في حقل طول المدي extent length للإشارة إلى امتداد من الكتل تتضمن بيانات بدون تهيئة uninitialized، وتفسر عند قراءتها ككتل صفرية zero-filled. وهذا يسمح بحجز مجموعة كتل متماسة من أجل ملف معين.

103. ^ أ، ب، ت، e2fsprogs (وتسمى أيضا e2fs programs) حزمة من الأدوات الضرورية في لينكس من أجل إنشاء ، وتفحص، وصيانة أنظمة ملفات ext2/3/4 على القرص. النسخة (e2fsprogs-1.44.1) تتضمن التالي:

أدوات	العديد من هذه الأدوات ترتكز على مكتبة libext2fs library / تنبيه : أي استعمال خاطئ سوف يتلف نظام الملفات !	
chattr (1)	change file attributes on a Linux file system	تغيير خصائص الملفات على نظام ملفات لينكس
lsattr (1)	list file attributes on a Linux second extended file system	سرد خصائص الملفات على نظام ملفات لينكس ext2
badblocks (8)	search a device for bad blocks	البحث عن الكتل المعيبة (التالفة) على القرص
debugfs (8)	used to manually view or modify internal structures of the file system	يستخدم في عرض أو تعديل هياكل نظام الملفات الداخلية ext2/ext3/ext4 file system debugger
dumpe2fs (8)	dump ext2/ext3/ext4 filesystem information	طباعة معلومات الكتلة العليا للنظام الملفات ومجموعة الكتل
e2freefrag (8)	report free space fragmentation information	تقدم تقريرا عن معلومات تجزئة المساحة الحرة
e2fsck (8)	fsck.ext2 (8) fsck.ext3 (8) fsck.ext4 (8) : check a Linux ext2/ext3/ext4 file system	إحدى أدوات fsck التي تتفحص وتصحح التضاربات في أنظمة ملفات لينكس ext2/ext3/ext4
e2image (8)	Save critical ext2/ext3/ext4 filesystem metadata to a file	يحفظ في ملف البيانات الوصفية الحرجة لأنظمة ملفات ext2/ext3/ext4
e2label (8)	Change the label on an ext2/ext3/ext4 filesystem	يغير لصيقة (اسم / عنوان) على أنظمة ملفات ext2/ext3/ext4
e2undo (8)	Replay an undo log for an ext2/ext3/ext4 filesystem	إعادة تشغيل سجل التراجع undo log لأنظمة ملفات ext2/ext3/ext4
e4crypt (8)	ext4 filesystem encryption utility	وسيلة تستخدم في تشفير نظام ملفات ext4
e4defrag (8)	online defragmenter for ext4 filesystem	برنامج لإلغاء التجزئة في وضع المتصل من أجل نظام ملفات ext4
filefrag (8)	report on file fragmentation	تقدم تقريرا عن تجزئة الملفات
logsave (8)	save the output of a command in a logfile	يحفظ خرج الأمر إلى ملف سجل
mke2fs (8)	mkfs.ext2 (8) mkfs.ext3 (8) mkfs.ext4 (8) : create an ext2/ext3/ext4 filesystem	يستخدم في إنشاء أنظمة ملفات ext2/ext3/ext4
mklost+found (8)	create a lost+found directory on a mounted Linux second extended file system	إنشاء دليل lost+found على ext2 موصول في لينكس (عادة lost+found يكون في الدليل الجذر)
resize2fs (8)	ext2/ext3/ext4 file system resizer	إعادة تحجيم (توسيع وتقليص) أنظمة ملفات ext2/ext3/ext4
tune2fs (8)	adjust tunable filesystem parameters on ext2/ext3/ext4 filesystems	يستخدم في ضبط وتعديل معاملات نظام الملفات

104. ^ أ، ب، ميزّات نظام ملفات ext4 (32 بت (قناع بت)):

- **مجموعة أعلام الميزّات المتوافقة Compatible feature set :**

نظام التشغيل يستطيع وصل (نظام الملفات) حتى وإن كان لا يدعم هذه الميزّات. وتطبيق نظام الملفات سيكون حر في دعم أو عدم دعم هذه الميزات دون خطر إتلاف البيانات الوصفية.

النواة ما زال بإمكانها القراءة / الكتابة إلى نظام الملفات حتى وإن لم تفهم العلم؛ لكن أداة fsck لا يجب أن تفعل ذلك.

- **مجموعة أعلام الميزّات الغير متوافقة Incompatible feature set :**

تطبيق نظام التشغيل لا يجب أن يصل (نظام الملفات) إذا كان لا يدعم هذه الميزّات. والتطبيق الذي لا يدعم هذه الميزات سيكون غير قادر على استخدام نظام الملفات بالشكل الصحيح. مثلا، إن كان ضغط البيانات مستخدم، بعد قراءة ملف تنفذي من القرص، سيكون غير صالح للاستعمال إن كان النظام لا يعرف فك ضغط الملف.

النواة أو أداة fsck يجب أن تتوقف إذا لم تفهم إحدى هذه بتات.

- **مجموعة أعلام الميزّات المتوافقة – في وضعية القراءة فقط Read-only compatible feature set:**

تطبيق نظام الملفات ينبغي أن يوصل (نظام الملفات) في وضعية القراءة فقط، إذا كانت الميزة المحددة بدون دعم.

النواة ما زال بإمكانها وصل نظام الملفات في وضعية القراءة فقط إذا لم تفهم إحدى هذه بتات.

- **مجموعة الميزّات التجريبية (الاختبارية)! Experimental feature set :**

يمكن أن تكون أي شيء مضاف إلى النواة.

وصف	دعم			نوع	ثابت / معامل	علم	رمز تذكري
	متوافق خلفيا	e2fsprogs-1.44.0	نواة				
التخصيص المسبق للكتل الدليل	√	√	ext4, 4,20	s_feature_compat	EXT2_FEATURE_COMPAT_DIR_PREALLOC	0x1	dir_prealloc
"imagic inodes"	√	√	ext4, 4,20		EXT2_FEATURE_COMPAT_IMAGIC_INODES	0x2	imagic_inodes
نظام ملفات مزود بقيد حوادث	√	√	ext3, 2.4.15		EXT3_FEATURE_COMPAT_HAS_JOURNAL	0x4	has_journal
دعم الخصائص الممتدة	√	√	ext2/ext3, 2.6.0		EXT2_FEATURE_COMPAT_EXT_ATTR	0x8	ext_attr
النظام يملك الكتل المحجوزة GDT	√	√	ext3, 2.6.10		EXT2_FEATURE_COMPAT_RESIZE_INODE	0x10	resize_inode (online resizing)
النظام يملك فهارس للدليل	√	√	ext3, 2.6.0		EXT2_FEATURE_COMPAT_DIR_INDEX	0x20	dir_index
من أجل مجموعة الكتل الغير مهيئة !	√	√	X		EXT2_FEATURE_COMPAT_LAZY_BG	0x40	lazy_bg
"exclude inode" (غير مستخدم في ext4) (مستخدم في Next3)		X	X		EXT4_FEATURE_ COMPAT_EXCLUDE_INODE	0x80	exclude inode
"Exclude bitmap" (تتعقب الكتل المعينة إلى snapshot files)	√	√	X		EXT2_FEATURE_COMPAT_EXCLUDE_BITMAP	0x100	snapshot_bitmap
تمكين ميزة النسختان sparse_super2		√	ext4, 3.16		EXT4_FEATURE_COMPAT_**SPARSE**_SUPER2	0x200	sparse_super2
دعم إنشاء وحذف الملفات المعزولة !		X	X		EXT4_FEATURE_COMPAT_ORPHAN_FILE	0x400	orphan_file
النظام يستخدم ضغط البيانات	√	√	ext4, 4,20	s_feature_incompat	EXT2_FEATURE_INCOMPAT_COMPRESSION	0x1	compression
نوع الملف مضمن في مدخلة الدليل	√	√	ext2, 2.2.0		EXT2_FEATURE_INCOMPAT_FILETYPE	0x2	filetype
النظام يحتاج إلى استعادة !	√	√	ext4, 4,20		EXT3_FEATURE_INCOMPAT_RECOVER	0x4	needs_recovery
النظام يملك جهاز قيد حوادث منفصل	√	√	ext4, 4,20		EXT3_FEATURE_INCOMPAT_JOURNAL_DEV	0x8	journal_dev
ميزة مجموعة الكتل الوصفية !	√	√	ext4, 2.6.28		EXT2_FEATURE_INCOMPAT_META_BG	0x10	meta_bg
الملفات تستخدم المديات	√	√	ext4, 2.6.28		EXT3_FEATURE_INCOMPAT_EXTENTS	0x40	extent / extents
تمكين حجم نظام الملفات 2^{64} كتلة		√	ext4, 2.6.28		EXT4_FEATURE_INCOMPAT_64BIT	0x80	64bit
حماية نظام الملفات من الوصل المتعدد		√	ext4, 3.0		EXT4_FEATURE_INCOMPAT_MMP	0x100	mmp
ميزة مجموعة الكتل المرنة !		√	ext4, 2.6.28		EXT4_FEATURE_INCOMPAT_FLEX_BG	0x200	flex_bg
قيم الخصائص الممتدة الكبيرة في inode		√	ext4, 4.13		EXT4_FEATURE_INCOMPAT_EA_INODE	0x400	ea_inode
بيانات في مدخلة الدليل		√	ext4, 4,20		EXT4_FEATURE_INCOMPAT_DIRDATA	0x1000	dirdata
بذرة تدقيق مجموع البيانات الوصفية في Superblock		√	ext4, 4.4		EXT4_FEATURE_INCOMPAT_CSUM_SEED	0x2000	metadata_csum_seed
دليل كبير > 2GB أو htree مستوى 3		√	ext4, 4.13		EXT4_FEATURE_INCOMPAT_LARGEDIR	0x4000	large_dir
تضمين بيانات في inode		√	ext4, 3.8		EXT4_FEATURE_INCOMPAT_INLINE_DATA	0x8000	inline_data
Inodes مشفرة في نظام الملفات		√	ext4, 4.1		EXT4_FEATURE_INCOMPAT_ENCRYPT	0x10000	encrypt
دعم ترميز المحارف على مستوى نظام الملفات		e2fsprogs 1.45.1	ext4, 5,2		EXT4_FEATURE_INCOMPAT_CASEFOLD	0x20000	casefold (fname_encoding)
توصيف المجموعات ونسخ الكتلة العليا متناثرة	√	√	ext2, 2.2.0	s_feature_ro_compat	EXT2_FEATURE_RO_COMPAT_SPARSE_SUPER	0x1	sparse_super
نظام الملفات يستخدم في تخزين الملفات الكبيرة	√	√	ext2, 2.2.0		EXT2_FEATURE_RO_COMPAT_LARGE_FILE	0x2	large_file
محتوى الدليل مخزن في شكل شجرة ثنائية (غير مستخدم)		X	X		EXT4_FEATURE_RO_COMPAT_BTREE_DIR	0x4	btree_dir
حجم الملف الكبير (بوحدات من الكتل المنطقية)		√	ext4, 2.6.28		EXT4_FEATURE_RO_COMPAT_HUGE_FILE	0x8	huge_file
توصيف المجموعات يملك تدقيق مجاميع		√	ext4, 2.6.28		EXT4_FEATURE_RO_COMPAT_GDT_CSUM	0x10	uninit_bg / uninit_groups
تجاوز حد الأدلة الثانوية 32,000 في ext3		√	ext4, 2.6.28		EXT4_FEATURE_RO_COMPAT_DIR_NLINK	0x20	dir_nlink
نظام الملفات يملك inodes كبيرة		√	ext4, 2.6.28		EXT4_FEATURE_RO_COMPAT_EXTRA_ISIZE	0x40	extra_isize
نظام الملفات يملك صور snapshot (غير مستخدم)		X	X		EXT4_FEATURE_RO_COMPAT_HAS_SNAPSHOT	0x80	snapshot
تمكين نظام الحصص	√	√	ext4, 3.6		EXT4_FEATURE_RO_COMPAT_QUOTA	0x100	quota
تخصيص الكتل الكبيرة		√	ext4, 3.2		EXT4_FEATURE_RO_COMPAT_BIGALLOC	0x200	bigalloc
دعم تدقيق مجموع البيانات الوصفية		√	ext4, 3.18		EXT4_FEATURE_RO_COMPAT_METADATA_CSUM	0x400	metadata_csum
دعم النسخ طبق الأصل		√	X		EXT4_FEATURE_RO_COMPAT_REPLICA	0x800	replica
صورة نظام ملفات للقراءة فقط		√	X		EXT4_FEATURE_RO_COMPAT_READONLY	0x1000	Read-only
		X	X		EXT4_FEATURE_RO_COMPAT_ORPHAN_PRESENT	0x2000	orphan_file_used
نوع جديد من أجل تعقب حصص القرص		√	ext4, 4.5		EXT4_FEATURE_RO_COMPAT_PROJECT	0x2000	project
تمكين مشاركة الكتل في نظام الملفات (في وضعية القراء فقط)		√	X		EXT4_FEATURE_RO_COMPAT_SHARED_BLOCKS	0x4000	shared_blocks
للتحقق من صحة الملفات		e2fsprogs v1.45.2	X		EXT4_FEATURE_RO_COMPAT_VERITY	0x8000	verity / fsverity

تنبيه : بعض الميزات (حتى وإن كانت قديمة!) ربما ما زالت في مرحلة الاختبار ولا ينصح باستخدامها إلا على أجهزة الاختبار. أيضا بعض الميزات الموجودة في e2fsprogs ليست بالضرورة موجودة أيضا في نواة لينكس (جدول خاص)

105. ^ الدليل lost+found الذي ينشأ آليا سوف يستخدم مؤشر الفهرسة 11، إذن أول عنوان inode في مساحة المستخدم (غالبا للاستعمال الشخصي) سيكون 12 في هذا المثال كان من أجل الملف textfile:

```
# ls -ila
total 32
     2 drwxr-xr-x   4 mete mete  4096 Aug 23 14:56 .
3932162 drwxr-xr-x 104 mete mete  4096 Aug 23 14:59 ..
    11 drwx------   2 mete mete 16384 Aug 23 11:20 lost+found
 521217 drwxrwxr-x   2 mete mete  4096 Aug 23 14:56 testdir
     12 -rw-rw-r--   1 mete mete     5 Aug 23 14:56 textfile
# stat testfile
  File: 'testfile'
  Size: 5         Blocks: 8          IO Block: 4096   regular file
Device: 810h/2064d Inode: 12        Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/mete)  Gid: ( 1000/mete)
Access: 2017-08-22 15:10:27.131878596 +0200
Modify: 2017-08-22 15:06:45.229604104 +0200
Change: 2017-08-22 15:06:45.229604104 +0200
Birth: -
```

". "رابط صلب يشير إلى الدليل الحالي، في هذا المثال كان الدليل الجذر

106. ^ هذا الحقل مهم إن كان مولد أرقام inode يستطيع توليد أرقام مختلفة لنفس الكائن في أوقات مختلفة. وهذا نادر في أنظمة ملفات القرص، لكن يمكن أن يحدث في أنظمة ملفات الشبكة (NFS)

107. ^ أ، ب، إذا كانت هذه القيمة 0 أو (1)=EXT3_JNL_BACKUP_BLOCKS، حقل s_jnl_blocks سوف يتضمن نسخة مطابقة من مصفوفة i_block[]. و i_size في inode

108. ^ 16 بت (لا تحمل إشارة) تشير إلى موقع قيمة هذه الخاصية على كتلة القرص حيث تخزن.(أي إزاحة بايت في الكتلة المحددة). بالنسبة لأنظمة لينكس الحالية، هناك كتلة واحدة فقط، ولا يتم تعيين حقل الكتلة. عدة خصائص يمكنها تشارك نفس القيمة. بالنسبة للخاصية attribute في inode هذه القيمة لها صلة ببداية أول مدخلة؛ أما بالنسبة للكتلة فهذه القيمة لها صلة ببداية الكتلة (أي التروسة).

109. ^ 32 بت تشير إلى قيمة هاش الخاصة hash value بقيمة الخاصية attribute value واسم الخاصية attribute name. النواة لن تحدد hash للخصائص الموجودة داخل Inode، في هذه الحالة، هذه القيمة يجب أن تكون صفر، لأن أداة e2fsck تتحقق من صحة أي hash ليست صفر بغض النظر عن مكان xattr.

110. ^ إذا كان نوع الخاصية user، أو trusted، أو security القيمة في نهاية الكتلة ستكون جزء من قيمة زوج الخاصية. وإذا كان أحد أنواع POSIX ACL، القيمة ستملك تشكيلتها الخاصة من هياكل البيانات.

111. ^ أ، ب، ت، ث، ج، أولا، الهوية الحقيقة لأي ملف ستكون رقم مؤشر الفهرسة inode number وليس اسم الملف، (الذي يمكن تغييره بسهولة)

الوصلة الصلبة أو الرابط الصلب Hard Link عبارة عن مدخلة دليل تربط بين الاسم والملف (تحديدا inode number) على نظام الملفات. جميع أنظمة الملفات التي ترتكز على الأدلة يجب أن تملك (على الأقل) رابط صلب واحد يمنح إلى الاسم الأصلي لكل ملف. تعبير "hard link" عادة يستخدم فقط في أنظمة الملفات التي تسمح بأكثر من رابط صلب مع نفس الملف.

عملية إنشاء رابط صلب تعني منح ملف واحد أسماء متعددة (أو نفس الاسم مكرر) في أدلة مختلفة وجميعها مستقلة لكنها مرتبطة بنفس البيانات على القرص. ولا أحد منها يعتمد على الأخر,

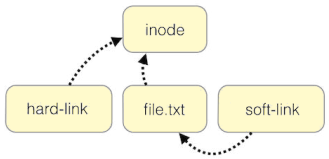
وهذا له تأثير الاسم المستعار أو الكنية؛ مثلا، إذا فتحنا الملف عبر أحد أسماءه، وتم تعديل المحتوى، التغييرات ستظهر عند فتح الملف من اسم بديل أخر، على خلاف، الوصلة الرمزية (أو المختصر) التي ليست رابط مباشر إلى البيانات نفسها. ولكنها بالأحرى ملف قصير يتضمن نص من اسم الملف،أو موقع يمنح منفذ مباشر إلى اسم ملف أخر ضمن دليل معين.

الاسم المضمن أو المشار إليه بواسطة الوصلة الرمزية يمكن أن يكون رابط صلب أو وصلة رمزية أخرى. هذا أيضا له تأثير aliasing، لكن بطريقة مختلفة.

ولأن inode بنية بينات تمثل كائن (مثل ملف!) في نظام الملفات، وترتبط داخليا بنظام الملفات، لذلك لا يمكن للروابط الصلبة hard-link الإشارة إلى inode على نظم الملفات الأخرى، عكس وصلات الرمزية (symbolic link, symlink, soft link). التي يمكنها الإشارة إلى الملفات على نظم الملفات الأخرى. الوصلة الرمزية تعني أي ملف يتضمن مرجع إلى ملف أو دليل أخر في شكل مسار نسبي أو مطلق الذي يأثر على ترجمة / تحليل اسم المسار pathname resolution.

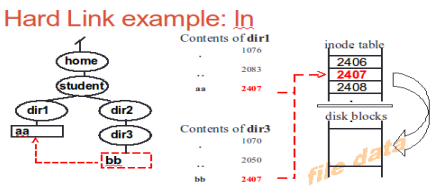
الوصلة الصلبة (مدخلة دليل)

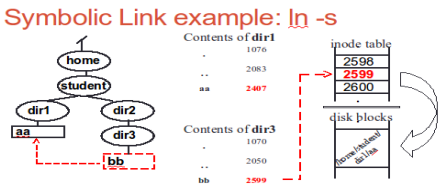
- كل inode يمثل ملف واحد يمكن أن يقترن بعدة أسماء (لا يخضع لاسم الملف).
- الروابط الصلبة يمكنها فقط الإشارة إلى الملفات. هذا يمنع الدورات cycles في شجرة الدليل (باستثناء روابط "." و"..").
- الروابط الصلبة تملك نفس أرقام مؤشر فهرسة inode numbers
- الروابط الصلبة تستخدم فقط داخل نظام الملفات الواحد.
- الروابط الصلبة لا تدعمها جميع أنظمة الملفات.
- تعداد الروابط i_links_count يزداد مع كل إنشاء رابط ويتناقص مع كل حذف للملف (مثل رابط) من الدليل. inode والبيانات المرتبطة تحذف فقط عندما يصبح تعداد المراجع 0. (حينذاك يستحيل الوصول إلى كتل البيانات)



الوصلة الرمزية (ملف خاص)

- الوصلة الرمزية هي ملف خاص يتضمن اسم ملف (يشير إلى ملف أخر) "مضمون" الملف هو هدف الوصلة
- الوصلة الرمزية يمكنها الإشارة إلى أي نوع من الملفات (بما في ذلك، الدليل)
- الوصلة الرمزية تملك أرقام مؤشر فهرسة مختلفة inode number
- الوصلة الرمزية يمكنها الإشارة إلى الملفات على نظم الملفات الأخرى
- إذا كان طول مسار وجهة الملف/ الدليل أقل من 60 محرف (fast symlink) يخزن في حقل 60 بايت داخل inode.(عادة، حقل 60 بايت حقل مخصص للتخزين extents أو عنونة الكتل المباشرة 12 و الغير مباشرة الثلاثة)
- إذا كان المسار أطول من 60 محرف، تخصص كتلة، لاحتواء مسار الوجهة. حجم الملف سيتماشى مع طول المسار.
- النواة عندما تصادف الوصلة الرمزية أثناء بحث اسم المسار تستبدل اسم الوصلة بمضمونها (اسم الملف الهدف)، وتعيد تشغيل ترجمة اسم المسار
- الوصلة الرمزية يمكن أن تصبح مؤشر مدلّى أو يتيم "orphaned" (لا يقود إلى أي مكان) إذا تم حذف ملف الهدف.
- الوصلة الرمزية تستخدم inodes أكثر من الروابط الصلبة (2 مقابل 1)
- وصلات الرمزية لا تأثر على i_links_count. وعندما يصل تعداد الروابط إلى 0 يتم تحرير inode والكتل المرتبطة
- وصلات الرمزية تملك فوقانية overhead أعلى من الروابط الصلبة من أجل ترجمة أو تحليل اسم المسار






```
# ls -li original_file your_symbolic_link your_hard_link
391252 -rw-rw-r-- 2 xxx xxx 15 Nov 23 19:25 original_file
391252 -rw-rw-r-- 2 xxx xxx 15 Nov 23 19:25 your_hard_link
391071 lrwxrwxrwx 1 xxx xxx 13 Nov 23 19:23 your_symbolic_link -> original_file
```

ما هي أسماء الملفات التي تشير إلى رقم inode (تنبيه: لا يمكن إيجاد جميع وصلات الرمزية إلى الملف، لأنها ستكون في أي مكان)

```
# find / -inum 391252
/home/xxx/Desktop/your_hard_link
/home/xxx/Desktop/original_file
```

رغم أنها نظريا ممكنة، محاولة إنشاء روابط صلبة إلى الأدلة سوف تفشل (لأنها تكسر بنية نظام الملفات، باستثناء في بعض أنظمة يونكس) :

```
# ln -d -F mydir mydirlink
ln: failed to create hard link `mydirlink' => `mydir': Operation not permitted
# link mydir mydirlink2
link: cannot create link `mydirlink2' to `mydir': Operation not permitted
```

```
-d, -F, --directory
allow the superuser to attempt to hard link directories (note: will probably fail due to system restrictions, even for the superuser)
```

عند إنشاء الرابط الصلب، لن يكون هناك أي اختلاف بين اسم الملف الأصلي والرابط الصلب (الاسم الثاني) باستثناء عدد الروابط الذي سوف يصبح 2

```
# ln  /Path/somedir/original_file  your_hard_link
# stat original_file
File: `original_file'
Size: 15        Blocks: 8          IO Block: 4096   regular file
Device: 820h/2080dInode: 391252    Links: 2
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     xxx)   Gid: ( 1000/     xxx)
[Removed]
# stat your_hard_link
File: `your_hard_link'
Size: 15              Blocks: 8          IO Block: 4096   regular file
Device: 820h/2080d    Inode: 391252      Links: 2
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     xxx)   Gid: ( 1000/     xxx)
[Removed]
```

عند إنشاء الوصلة الرمزية الاختلاف سيكون في رقم inode ونوع الملف، والحجم هو عدد بايتات في اسم الملف المشار إليه، والأذون ستكون مفتوحة.

```
# ln -s /Path/somedir/original_file  your_symbolic_link
# stat your_symbolic_link
File: `your_symbolic_link' -> `original_file'
  Size: 13              Blocks: 0          IO Block: 4096   symbolic link
Device: 820h/2080d      Inode: 391071      Links: 1
Access: 0777/lrwxrwxrwx  Uid: ( 1000/     xxx)   Gid: ( 1000/     xxx)
[Removed]
# stat original_file
File: `original_file'
Size: 15              Blocks: 8          IO Block: 4096   regular file
Device: 820h/2080d    Inode: 391252      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     xxx)   Gid: ( 1000/     xxx)
[Removed]
```

112. ^ أ، ب، ت، إصلاح الكتلة المتضررة super block باستخدام البديلة (الاحتياطية) Alternative Superblocks (في المثال ext4 كان في القسم sda1) الخطوات التالية بعد الإقلاع عبر قرص التنصيب Live CD ثم من الطرفية Terminal:

```
# fdisk -l  (تحديد قسم لينكس على القرص)
# umount /dev/sda1 (فصل نظام الملفات/ إلغاء نقطة الارتباط)
# fsck.ext4 -v /dev/sda1 (تأكد من المشكلة)
# mke2fs -n /dev/sda1 (تحديد موقع النسخ البديلة / الاحتياطية)
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

# e2fsck -f -b 98304 /dev/sda1 (استعادة الكتلة العليا من النسخة الاحتياطية)
```

حاول استعادة superblock من النسخة الاحتياطية، مع إعادة تشغيل، إذا فشلت هذه الخطوة واصل تكرار العملية باستخدام رقم كتلة مختلف حتى تتأكد من إصلاح الكتلة :

الخيار f- يعني عمل فحص إجباري للنظام الملفات. والخيار b- يشير إلى alternative superblock، إذا فشلت الخطوة 4، حاول تكرارها مع القيمة في الخطوة 3

أيضا يمكن وصل النظام باستخدام الكتلة البديلة، بإنشاء نقطة وصل ثم وصل النظام عن طريق الكتلة 98304

```
# mkdir -p /mnt/linux1 (إنشاء نقطة وصل)
# mount -o sb=98304 /dev/sda1 /mnt/linux1  (وصل نظام الملفات عن طريق الكتلة 98304)
```

أو فحص كامل نظام الملفات والكتل المعيبة bad blocks (باستخدام طرفية قرص توزيعة لينكس LiveCD):

```
# umount /dev/sda1 (فصل نظام الملفات/ إلغاء نقطة الارتباط)
# fsck.ext4 -fck /dev/sda1 (فحص كامل نظام الملفات والكتل المعيبة)
```

تنبيه : أحيانا بعض الميزات (خصوصا التجريبية منها) في نظام الملفات تمنع إقلاع النظام والحل السريع طبعا سيكون في تعطيل الميزة.

113. ^ أ، ب، ت، ث، ج، إيجاد كتل البيانات في قيد الحوادث journal (مثال للمقارنة والتحليل) أولا، رقم journal inode سيكون "8"، للتأكد من ذلك راجع معلومات Super block :

```
# dumpe2fs -h /dev/sda1 | grep -i "Journal inode"
Journal inode:            8
```

استخراج كتل journal inode

```
# debugfs /dev/sda1
debugfs:  stat <8>
Inode: 8   Type: regular    Mode:  0600   Flags: 0x80000
Generation: 0    Version: 0x00000000:00000000
User:     0   Group:     0   Size: 134217728
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 262144
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x4e1b45e2:00000000 -- Mon Jul 11 12:50:10 2011
 atime: 0x4e1b45e2:00000000 -- Mon Jul 11 12:50:10 2011
 mtime: 0x4e1b45e2:00000000 -- Mon Jul 11 12:50:10 2011
crtime: 0x4e1b45e2:00000000 -- Mon Jul 11 12:50:10 2011
Size of extra inode fields: 28
EXTENTS:
(0-32766):6324224-6356990, (32767):6356991
```

كتلة نظام الملفات block #3 عند journal block 13972 وكتلة journal superblock عند 6324224، إذن على القرص (13972 + 6324224) = 6338196 :

```
# debugfs -c -R "logdump -a" /dev/vg_mookie/lv_root
debugfs 1.42.13.wc5 (15-Apr-2016)
Journal starts at block 13970, transaction 1103250
Found expected sequence 1103250, type 1 (descriptor block) at block 13970
Dumping descriptor block, sequence 1103250, at block 13970:
  FS block 12058640 logged at journal block 13971 (flags 0x0)
  FS block 3 logged at journal block 13972 (flags 0x2)
  FS block 12059015 logged at journal block 13973 (flags 0x2)
  FS block 12058703 logged at journal block 13974 (flags 0x2)
  FS block 12066941 logged at journal block 13975 (flags 0x2)
  FS block 12058641 logged at journal block 13976 (flags 0x2)
  FS block 12059178 logged at journal block 13977 (flags 0x2)
  FS block 0 logged at journal block 13978 (flags 0xa)
Found expected sequence 1103250, type 2 (commit block) at block 13979
```

```
# dd if=/dev/vg_mookie/lv_root count=1 bs=4096 skip=3 | od -Ax -tx4 | head
000000 00800000 00800010 00800020 0751583d
000010 00040694 00000000 00000000 112c0000
000020 00800001 00800011 00800220 1c581692
000030 00040056 00000000 00000000 aae40000
000040 00800002 00800012 00800420 191b214e <--this is modified in journal (هذه القيمة معدلة في قيد الحوادث)
000050 000400a5 00000000 00000000 31771575
000060 00800003 00800013 00800620 20006380
000070 00050000 00000000 00000000 fa6f2000
000080 00800004 00800014 00800820 20006d83
000090 00050000 00000000 00000000 7fff2000
```

```
# dd if=/dev/vg_mookie/lv_root count=1 bs=4096 skip=6338196 | od -Ax -tx4 | head
000000 00800000 00800010 00800020 0751583d
000010 00040694 00000000 00000000 112c0000
000020 00800001 00800011 00800220 1c581692
000030 00040056 00000000 00000000 aae40000
000040 00800002 00800012 00800420 191b214f <--is 0x214e in filesystem (القيمة في نظام الملفات )
000050 000400a5 00000000 00000000 cd261575
000060 00800003 00800013 00800620 200063a9
000070 00050000 00000000 00000000 aeb72000
000080 00800004 00800014 00800820 20006d83
000090 00050000 00000000 00000000 7fff2000
```

تقريبا الكتلتان متشابهتان، باستثناء بايت 0x4f، الذي يملك في قيد الحوادث، قيمة متغيرة من 0x4e إلى 0x4f.

وفقا للأمر "dumpe2fs -x"، الكتلة block #3 هي كتلة group descriptor block، التي تصف المجموعات 256-377. والقيمة (0x214e = 8526) هي تعداد الكتل الحرة في المجموعة 258

```
Group 258: (Blocks 0x810000-0x817fff) [ITABLE_ZEROED]
  Checksum 0x3177, unused inodes 5493
  Block bitmap at 0x800002 (bg #256 + 2), Inode bitmap at 0x800012 (bg #256 + 18)
  Inode table at 0x800420-0x80061f (bg #256 + 1056)
  8526 free blocks, 6427 free inodes, 165 directories, 5493 unused inodes
```

في مجموعة الكتل هذه، يبدوا أن بعض الكتل قد تم تخصيصها أو تحريريها، أيضا هناك كتل في قيد الحوادث يمكنها تغيير هذه القيمة. (هذا المثال منقول عن Andreas Dilger)

114. ^ أ، ب، إنشاء جهاز قيد حوادث خارجي external journal device (في نظام ملفات ext4)

أولا، لا يمكنك دائما الاعتماد على journal، خصوصا في حالة فشل العتاد والحل دائما في النسخ الاحتياطي الدوري للملفات المهمة (على وسيط خارجي) أيضا حجم جهاز قيد الحوادث الخارجي يجب أن يكون على الأقل 1024 × حجم كتلة نظام الملفات وحجم الكتلة في جهاز قيد الحوادث الخارجي يجب أن يكون بنفس حجم الكتلة في نظام الملفات. ووفقا للكاتب المدونة، التحسن في أداء النظام سيقارب 40% باستخدام جهاز قيد الحوادث الخارجي (على افتراض، أن جهاز قيد الحوادث يقع في قرص فيزيائي منفصل). أيضا نواة لينكس و e2fsck) حاليا لا تدعم قيود الحوادث الخارجية المشتركة رغم وجود دعم لربط أنظمة ملفات متعددة بجهاز قيد حوادث خراجي واحد.

فكرة نظام الملفات المزود بقيد حوادث، تقوم أساسا على تسجيل التغييرات في نظام ملفات (بكتابة البيانات مسبقا write-ahead) إلى قيد (سِجِلّ) هذا الأخير سيكون سجل دوري ومحجوز في منطقة محددة، قبل كتابة البيانات فعليًا إلى نظام الملفات (عبر checkpointing). بهذه الطريقة، إنشاء وحذف وتعديل الملف يصبح إجراء- وتطبق هنا نفس فكرة ذرية الإجراءات الموجودة في نظم إدارة قواعد البيانات DBMS لكن ليس بنفس التعقيد (لا حاجة إلى concurrency و isolation)، الذرية هنا تعني إما تحديث جميع البيانات الوصفية للكائن (ملف) أو لا يتم تحديثها إطلاقا. في هذه الحالة حدوث أي فشل في نظام الملفات، يعني أن هذا الأخير سيحاول العودة إلى الوضعية الصحيحة قبل حدوث الفشل/ الخطأ. لكن لذلك ثمن ! لأن كتابة نظام ملفات أولا إلى قيد الحوادث قبل كتابة البيانات فعليا إلى الملف سيكون له تأثير سلبي (إلى حد ما) على أداء نظام الملفات.

أما الأسباب التي قد تدفعك لإنشاء جهاز قيد حوادث خارجي external journal device في ext4 فهي:

- تجنب فساد/تلف البيانات corruption في قيد الحوادث نفسه، عن طريق حفظ السجل في مكان آخر غير المكان الأصلي.
- إنشاء جهاز قيد حوادث خارجي منفصل، سينتج عنه تحسن ملحوظ في الأداء. (لأن نظام قيد الحوادث لن يكتب البيانات مرتين في النظام الأصلي).
- أيضا إنشاء journal على HDD بعيدا عن نظام الملفات الموجود في SSD مفيد لهذا الأخير، لأنه سيخفض من دورات الكتابة-وإعادة الكتابة الغير ضرورية write-rewrite cycles على خلايا كتل SSD

خطوات وخيارات إنشاء جهاز قيد حوادث خارجي باستخدام mke2fs(8) أو tune2fs(8) :

إنشاء جهاز قيد حوادث خارجي external journal device (مثلا في القسم /dev/sdb1 على قرص مفصل) ثم ربط نظام الملفات (مثلا الموجود على /dev/sda1) بذلك الجهاز:

```
# mke2fs -O journal_dev /dev/block_device_name ( إنشاء جهاز قيد حوادث خراجي )
# mke2fs -t ext4 -J device=/dev/journal_device_name ( ربط نظام الملفات بجهاز قيد الحوادث الخارجي )
# mke2fs -t ext4 -J device=/dev/journal_device_name  /dev/block_device_for_new_fs ( أو إنشاء نظام الملفات واختيار جهاز قيد الحوادث الخارجي في نفس الوقت )
# umount /dev/existing_fs ( أو تغيير قيد الحوادث الداخلي إلى خارجي، في نظام ملفات موجود فعليا على القرص، لكن تحتاج أولا إلى فصل نظام الملفات الحالي ثم تنفيذ بقية الأوامر )
# tune2fs -O ^has_journal /dev/blk_dev_for_existing_fs ( إلغاء قيد الحوادث الموجود )
# tune2fs -o journal_dev -j -J device=/dev/device_name /dev/blk_dev_for_existing_fs (ربط نظام الملفات بجهاز قيد حوادث تم إنشاءه سابقا)
```

أخيرا، لا تنسى تمكين خيار الوصل journal_async_commit ؛ بدون هذا الخيار، قسم ext4 في النهاية سيصبح في وضعية القراءة فقط read-only. قد يحدث ذلك بعد ساعات من التشغيل أو عاجلا بعد عمليات كثيفة من IO.

.115 ^ أ، ب، ت، ث، ج، ح طريقة إيجاد inode (في ext4)، لكن قبل ذلك، نحتاج إلى إنشاء ملف من أجل هذا الاختبار:

```
# echo Here is a new file >testfile
# ls -li testfile
389350 -rw-rw-r-- 1 xxx xxx 19 Oct 30 05:40 testfile

# istat /dev/sda1 389350
inode: 389350
Allocated
Group: 48
Generation Id: 1711068283
uid / gid: 1000 / 1000
mode: rrw-rw-r--
Flags:
size: 19
num of links: 1
Inode Times:
Accessed:Tue Oct 30 05:40:06 2018
File Modified:Tue Oct 30 05:40:06 2018
Inode Modified:Tue Oct 30 05:40:06 2018
Direct Blocks:
127754
```

بالمناسبة، يمكنك عن طريق رقم inode إيجاد اسم الملف : find / -inum <number>

أولا، حتى نتوصل إلى inode المطلوب 389350 نحتاج إلى بعض المعلومات من كتلة superblock وجدول توصيف مجموعة الكتل group descriptor table :

```
$ fsstat /dev/sda1
[...]
Block Size: 4096
Inodes per group: 8096
[...]
Group: 48:
Inode Range: 388609 - 396704
[...]
Inode Table: 1572896 - 1573401
[...]
```

مؤشر الفهرسة inode المطلوب كان في مجموعة 48 لاحظ في خرج fsstat أن العنوان المطلوب 389350 يقع داخل نطاق عدد inodes لكل مجموعة، (388609 – 396704)

نعلم أن حجم inode في ext4 هو 256 بايت، وحجم الكتلة 4096 بايت، هذا يعني أن هناك 16 مؤشر فهرسة لكل كتلة، في هذا المثال كان هناك 8096 مؤشر فهرسة لكل مجموعة، (8096 ÷ 16 = 506) إذن عدد كتل مؤشرات الفهرسة لكل مجموعة هو 506. لاحظ أن inode table في المجموعة 48 يحتل 506 كتلة من 1572896 – 1573401. لكن أين تقع كتلة مؤشر الفهرسة 389350 ؟

عنوان أول مؤشر فهرسة في المجموعة 48 كان 388609. بطرح هذه القيمة من 389350 نصل إلى مؤشر الفهرسة 741 من بداية جدول مؤشرات الفهرسة inode table.

```
# dd if=/dev/sda1 skip=1572896 bs=4096 count=506 | dd skip=741 bs=256 count=1 | hexdump -Cv
       00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0500 b4 81 e8 03 13 00 00 00  a6 e0 d7 5b a6 e0 d7 5b |...........[...[|
0510 a6 e0 d7 5b 00 00 00 00  e8 03 01 00 08 00 00 00 |...[............|
0520 00 00 08 00 01 00 00 00  0a f3 01 00 04 00 00 00 |................|
0530 00 00 00 00 00 00 00 00  01 00 00 00 05 83 18 00 |................|
0540 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0550 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0560 00 00 00 00 7b d4 fc 65  00 00 00 00 00 00 00 00 |....{..e........|
0570 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0580 1c 00 00 00 8c db 9f 58  8c db 9f 58 90 6f c3 55 |.......X...X.o.U|
0590 a6 e0 d7 5b 90 6f c3 55  00 00 00 00 00 00 00 00 |...[.o.U........|
05a0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
05b0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
05c0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
05d0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
05e0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
05f0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0600 a4 81 00 00 e8 05 00 00  49 13 cc 5b 04 f5 8a 58 |........I..[...X|
[Removed]
```

بايتات من 4 إلى 7 ـ (32-بت المنخفضة) تشير إلى حجم الملف، والقيمة 0x000013 تعني 19 بايت، 60 بايتات من 40 إلى 99 تتضمن معلومات شجرة المديات extents. (تذكر أن: ext4 يتعقب مضمون الملفات باستخدام المديات وليس مؤشرات الكتل block pointers)، بنية المدى extent ستكون بحجم 12 بايت، إذن هناك 5 مديات في كل inode، لكن أول 12 بايت من مساحة المديات (بايتات من 40-51) تحتلها بنية ترويسة المديات extent header، إذن العدد الفعلي للمديات extents في inode سيكون 4. (مضمون ترويسة المديات يظهر في الجدول أعلاه) :

بايتات من 40 إلى 41 تشير إلى الرقم السحري (التوقيع) (0xF30A = 62218) هذا الرقم للتمييز بين مختلف تطبيقات المديات. بسبب إضافة ميزّات جديدة. التوقيع قد يتغير للضمان التوافق الخلفي ـ

بايتات من 42 إلى 43 تشير إلى عدد المديات extents، و(0x0001 = 1) تعني امتلاك الملف مدى واحد فقط. بايتات من 44 إلى 45 تشير إلى العدد الأقصى للمديات، و(0x0004 = 4)، يعني أن العدد الأقصى 4 مديات داخل inode

بايتات من 46 إلى 47 تشير إلى عمق الشجرة (0x0000 = 0)، بايتات من 48 إلى 51 تشير إلى هوية أو رقم توليد الشجرة Generation ID ـ (0x00000000 = 0)

سوف نناقش "Depth of tree" و "Generation ID" في المقالات القادمة. بايتات من 52 إلى 55 تشير إلى رقم الكتلة المنطقية Logical block number ـ (0x00000000)، الذي يشير إلى مكان هذا المدى النسبي إلى بداية الملف وهذا سيكون مهم جدا في حال وجود العديد من المديات. لكن بما أننا نملك مدى واحد في هذا الملف، يجب أن يبدأ من بداية الملف أي من الكتلة المنطقية صفر

بايتات من 56 إلى 57 تشير إلى عدد الكتل في هذا المدى (0x0001) هذا الملف صغير، لذلك احتاج فقط إلى كتلة واحدة.

6 بايت التالية تشير إلى رقم الكتلة الفيزيائية من أول كتلة في المدى، أي البداية الفعلية للمدى على القرص :

بايتات من 58 إلى 59 تشير إلى عنوان الكتلة الفيزيائية physical block (16-بت العليا) (0x0000) بايتات من 60 إلى 63 تشير إلى عنوان الكتلة الفيزيائية (32-بت المنخفضة) (0x003A883F)

في أنظمة الحاسوب الحديثة القيم تكون محاذية للحدود 16-بت، أو 32-بت، أو 64-بت وعنوان كتلة 48-بت، سوف يمثل في قيمتين ؛ أول 2 بايت تشير إلى 16-بت العليا من عنوان الكتلة و 4 بايت تتضمن 32-بت المنخفضة من العنوان.

وبناء على ذلك، في هذا المثال، نترجم عنون الكتلة إلى 0x000000188305 (ست عشري) الذي يساوي رقم الكتلة 1606405 (عشري) أولا، دعنا نتأكد من ذلك :

```
# blkcat /dev/sda1 1606405 | hexdump -C
0000 48 65 72 65 20 69 73 20 61 20 6e 65 77 20 66 69 |Here is a new fi|
0010 6c 65 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 |le..............|
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[Removed]
```

بما أنه لا توجد مديات إضافية (بالنسبة للترويسة المديات السابقة)، 36 بايت التالية في inode ستكون شاغرة null.. الآن لاحظ ماذا يحدث داخل inode وكتلة / كتل البيانات عند حذف الملف testfile :

```
# rm testfile
# blkcat /dev/sda1 1606405 | hexdump -C
0000 48 65 72 65 20 69 73 20 61 20 6e 65 77 20 66 69 |Here is a new fi|
0010 6c 65 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 |le..............|
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[Removed]
```

كما ترى، كتلة أو كتل البيانات ما زالت موجودة بعد حذف الملف. وهذا هو السلوك المعياري في أنظمة الملفات. لكن ماذا عن inode :

 # dd if=/dev/sda1 skip=1572896 bs=4096 count=506 | dd skip=741 bs=256 count=1 | hexdump -Cv

مضمون Inode قبل حذف الملف

```
          00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 b4 81 e8 03 13 00 00 00  a6 e0 d7 5b a6 e0 d7 5b |...........[...[|
0010 a6 e0 d7 5b 00 00 00 00  e8 03 01 00 08 00 00 00 |...[............|
0020 00 00 08 00 01 00 00 00  0a f3 01 00 04 00 00 00 |................|
0030 00 00 00 00 00 00 00 00  01 00 00 00 05 83 18 00 |................|
0040 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0050 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0060 00 00 00 00 7b d4 fc 65  00 00 00 00 00 00 00 00 |....{..e........|
0070 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|

0000 b4 81 e8 03 00 00 00 00  a5 39 d9 5b a8 39 d9 5b |.........9.[.9.[|
0010 a8 39 d9 5b a8 39 d9 5b  e8 03 00 00 00 00 00 00 |.9.[.9.[........|
0020 00 00 08 00 01 00 00 00  0a f3 00 00 04 00 00 00 |................|
0030 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0040 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0050 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
0060 00 00 00 00 7b d4 fc 65  00 00 00 00 00 00 00 00 |....{..e........|
0070 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
[Removed]
```

مضمون inode بعد حذف الملف :

ماذا حدث في بنية inode عند إلغاء التخصيص deallocated. ؟. لاحظ بعد الحذف، إلى جانب تغير قيم التاريخ الأخرى ظهور 0x5BD939A8 الذي يمكن ترجمته باستخدام الطرفية: date -d @1540962728 ويعني تاريخ الحذف: Wed Oct 31 06:12:08 CET 2018 قيمة حجم الملف أصبح صفر. قيمة عدد المديات في ترويسة المديات extent header كذلك صفر.

المدى extent نفسه سوف يمسح (يصفر). مسح المدى extent يعني خسارة عنوان الكتلة الفيزيائية physical block address من أول الكتلة بالإضافة إلى طول المدى.

هذا يعني أنه لم تتبقى أية بيانات وصفية في inode يمكن أن تساعدنا في استعادة الملف المحذوف.هذا السلوك يشبه في ext3 مسح مؤشرات الكتل block pointers في inode عند إلغاء تخصيص inode للأسف، هذا يعني أننا مجبرين على استخدام الطرق التقليدية في استعادة الملفات traditional file-carving methods.

116. ^ أ، ب، ت، ث، ج، مدخلة inode في جدول مؤشرات الفهرسة inode table (هذا المثال في ext3)

في حقل أنماط الملف، قيمة 16-بت ستكون مجزأة إلى ثلاثة أقسام : في 9 بت المنخفضة توجد أعلام الأذون (0x100, 0x080, 0x040, 0x020, 0x010, 0x008, 0x004, 0x002,0x001)، حيث كل بت يقابل إذن. الأذون تستخدم مفهوم user، و group، و others (أو world)، في inode "المستخدم" يعني user ID، و"مجموعة" تعني group ID، و" الآخرون "others تعني هوية جميع المستخدمين الآخرون.

كل مجموعة من هذه المجموعات يمكن أن تملك إذن قراءة read و/أو كتابة write و/أو تنفيذ execute. أعلام هذه الأذون ستكون في جدول inode table. في بتات من 0 إلى 8 (الجدول أعلاه).

بتات من 9 إلى 11 (0x800, 0x400, 0x200) ستكون من أجل الأدلة والملفات التنفيذية executable files. إذا تم تعيين إحداها، سلوك الملف التنفيذي سيكون مختلف عند التشغيل، أو الملفات في الدليل سيكون لها خواص خاصة properties. بتات الأخيرة، من 12 إلى 15 للتعريف بنوع الملف (0xC000, 0xA000 , 0x8000 , 0x6000 , 0x4000 , 0x2000 , 0x1000) الذي من أجله تم إنشاء inode. في الحقيقة هذه قيم وليست أعلام، لذلك يعين واحد منها فقط.

في هذا المثال نتفحص مدخلة inode 16. لكن أولا نحتاج إلى تحديد هوية المجموعة التي ينتمي إليها inode. في هذا النظام 16,288 مؤشر فهرسة لكل مجموعة كتل، إذن نحن نبحث عن/في المجموعة 0.

```
$ echo "(16-1)/16288" | bc
0
```
(تساوي = 0)

عنوان بداية inode table يحدده توصيف المجموعة group descriptor، الذي كان عند الكتلة 4، وحجم الكتلة في نظام الملفات 4,096 بايت. لاستخراج البيانات، نستخدم أداة dd مع حجم الكتلة 4096 بايت من أجل القفز إلى inode table ثم نستخدم dd (مرة أخرى) مع حجم الكتلة 128 بايت للقفز إلى inode 16. (معلومة: أول inode سيكون 1، وليس 0 لذلك سنطرح 1 من قيمة القفزة) :

```
# dd if=ext3.dd bs=4096 skip=4 | dd bs=128 skip=15 count=1 | xxd
          0001 0203 0405 0607 0809 0A0B 0C0D 0E0F  0123456789ABCDEF
0000 a481 f401 0040 9c00 a4f0 2a4f f607 2a3f |.....@..m.*?..*?|
0016 8107 2a3f 0040 0000 f401 0100 404e 0000 |..*?.....@..@N..|
0032 0000 0000 0000 0000 2a38 8089 2d38 0000 |........*8..-8..|
0048 2e38 0000 2f38 0000 3038 0000 3138 0000 |.8../8..08..18..|
0064 3238 0000 3338 0000 3438 0000 3538 0000 |28..38..48..58..|
0080 3638 0000 3738 0000 3838 0000 393c 0000 |68..78..88..9<..|
0096 0000 0000 ba94 ea0b 0000 0000 0000 0000 |................|
0112 0000 0000 0000 0000 0000 0000 0000 0000 |................|
```

في هذا الطرح الست عشري، بايتات من 0 إلى 1 تشير إلى نمط الملف، الذي كان 0x81a4. هذه بتات تأكد لنا التالي: كل شخص يمكنه قراءة هذا الملف 0x004، ومجموعة المستخدمين group يمكنها قراءة هذا الملف 0x020، والمستخدم user يمكنه الكتابة إلى هذا الملف 0x080، والقراءة 0x100. بينما بتات العليا تشير إلى أن الملف هو ملف اعتيادي regular file ـ 0x8000 (بدون حساب علم الملف القيمة في الثماني ستكون 0644، أي -rw-r--r--)

(A481 = 10000001 10100100) = (100 = 0x04), (100000 = 0x020), (10000000 = 0x080), (100000000 = 0x100), (1000000000000000 = 0x8000)

بايتات من 2 إلى 3 تشير إلى رقم هوية المالك، الذي كان 500 و بايتات من 24 إلى 25 تشير إلى رقم هوية المجموعة، التي كانت 500. بايتات من 4 إلى 7 تشير حجم الملف 10,240,000 بايت (0x009c4000). بايتات من 8 إلى 11 تشير إلى زمن النفاذ 0x3f2a096d، الذي يترجم إلى August 1, 2003 at 06:32:13 UTC. (عن طريق: date -d @1059719533 $) بايتات من 12 إلى 15 ومن 16 إلى 19 ومن 20 إلى 23 تشير إلى بقية الأختام الزمنية. بايتات من 26 إلى 27 تشير إلى تعداد الروابط link count، الذي كان 1، هذا يعني أن هناك اسم ملف file name يشير إليه.

بايتات من 28 إلى 31 تشير إلى تعداد القطاعات أو الكتل sector count. بايتات من 32 إلى 35 تشير إلى عدم تعيين أية اعلام خاصة أو خصائص attributes. بايتات من 36 إلى 39 غير مستخدمة في ext3

بايتات من 40 إلى 43 من أجل مؤشر الكتلة المباشر(ة) الأولى first direct block pointer، الذي يشير إلى الكتلة 14,380 (0x0000382c). بايتات من 44 إلى 47 من أجل مؤشر الكتلة المباشر(ة) الثاني second direct pointer، الذي يشير إلى الكتلة 14,381 (0x0000382d). بايتات من 88 إلى 91 تتضمن عنوان مؤشر الكتلة الغير مباشر(ة) الفذ single indirect block pointer، الذي كان في الكتلة 14,392 (0x00003838). أيضا بايتات من 92 إلى 95 تشير إلى مؤشر الكتلة الغير مباشر(ة) المزدوج double indirect block pointer في الكتلة 15,417 (0x00003c39)

مضمون هاتين الكتلتين سيكون لائحة من عناوين 4-بايت. مؤشر الكتلة الغير مباشر(ة) الفذ single indirect block pointer يتضمن لائحة من عناوين حيث يتم تخزين مضمون الملف.

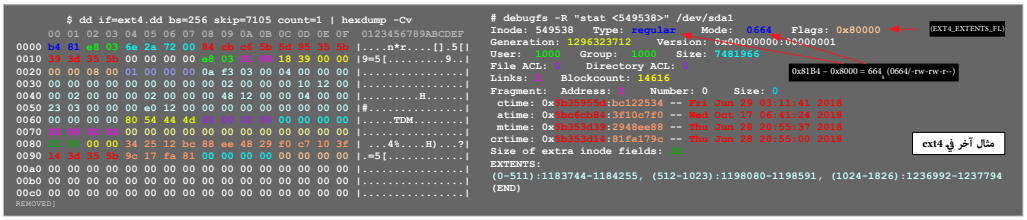
```
# blkcat -f linux-ext3 ext3.dd 14392 | xxd
          0001 0203 0405 0607 0809 0A0B 0C0D 0E0F  0123456789ABCDEF
0000 3938 0000 3a38 0000 3b38 0000 3c38 0000 |98..:8..;8..<8..|
0016 3d38 0000 3e38 0000 3f38 0000 4038 0000 |=8..>8..?8..@8..|
0032 4138 0000 4238 0000 4338 0000 4438 0000 |A8..B8..C8..D8..|
[REMOVED]
```

حالة تخصيص allocation status مؤشر الفهرسة تخزن في inode bitmap التي في نفس المجموعة وينتمي إليها inode. و group descriptor يتضمن عنوان كتلة inode bitmap، التي في المثال كانت الكتلة 3، وكان من مضمونها التالي:

```
# blkcat -f linux-ext3 ext3.dd 3 | xxd
          0001 0203 0405 0607 0809 0A0B 0C0D 0E0F  0123456789ABCDEF
0000 ff7f fcff 1f00 0000 00c8 0000 0000 0000 |................|
```

من أجل تحديد بايت الصحيح، نطرح 1 لبلوغ بداية أول inode في المجموعة ثم نقسم على 8. أي: (16 -1) / 8 =1 (البقية 7، تذكر أن 1 بايت = 8 بت إذن 16 inode في بايت f7). بت 16 inode كان البت الأكثر أهمية 11110111 في بايت 1 (0xf7)، الذي هو 1، إذن هو مخصص Allocated. تقريبا جميع معلومات inode 16 السابقة يمكن الحصول عليها بسهولة بالأمر istat :

```
# istat -f linux-ext3 ext3.dd 16
inode: 16
Allocated
Group: 0
Generation Id: 199922874
uid / gid: 500 / 500
mode: -rw-r--r--
size: 10240000
num of links: 1
Inode Times:
Accessed: Fri Aug 1 06:32:13 2003
File Modified: Fri Aug 1 06:24:01 2003
Inode Modified: Fri Aug 1 06:25:58 2003
Direct Blocks:
14380 14381 14382 14383 14384 14385 14386 14387
14388 14389 14390 14391 14393 14394 14395 14396
[REMOVED]
16880 16881 16882 16883
Indirect Blocks:
14392 15417 15418 16443
```



راجع شرح بقية العناصر في جدول بنية مدخلة struct ext4_inode أعلاه

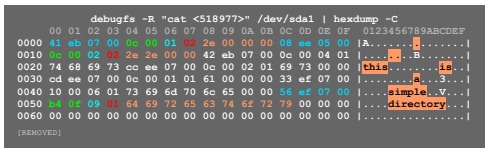
117. ^ أ، ب، ت، ث، ج، ح، مدخلات الدليل : ذكرنا سابقا، أن الدليل هو المكان الوحيد (في أنظمة ملفات يونكس التقليدية)، الذي تخزن فيه أسماء الملفات (اسم الكائن)، وأن الأدلة ملفات خاصة تربط أسماء الملفات بأرقام مؤشرات الفهرسة inode numbers <-- file names. وأن الأدلة في بنيتها البسيطة، مجرد لوائح متتالية من مدخلات الملفات. بدون ترتيب. وأن بنية مدخلة الدليل تشير إلى البيانات الوصفية (inode) التي بدورها تتضمن خصائص الملف وتشير إلى مضمون الملف (أي الكتل). وأن بنية مدخلة الدليل ستكون بإحدى الصيغتين، وكلاهما يملك نفس الحجم (راجع الجداول أعلاه)، وأن القيمة FILETYPE في الميزّات الغير متوافقة (في superblock) ستحدد الصيغة المستخدمة. (حاليا الصيغة الثانية). عموما، مدخلات الدليل في EXT تضاف إلى ملف الدليل (أي الدليل) بالترتيب الذي تنشأ فيه الملفات داخل الدليل. مثال على ذلك، سوف ننشئ دليل صغير باسم **testing** ، ثم داخل ذلك الدليل، سننشئ الملفات الخمسة التالية بهذا الترتيب: " this", "is" , "a" , "simple" , "directory"

```
$ mkdir testing
$ cd testing
$ touch this is a simple directory
```

لاحظ كيف تعرض أداة ls أسماء الملفات بالترتيب الأبجدي داخل الدليل testing

```
$ ls
a  directory  is  simple  this
```

لكن عند تفحص المضمون باستخدام debugfs و cat وتمرير خرج إلى hexdump تظهر مدخلات الدليل بترتيبها الفعلي:



على اليمين، تظهر أسماء الملفات (بمفرة أسكي) وبالترتيب الذي نشأت فيه الملفات في ملف الدليل

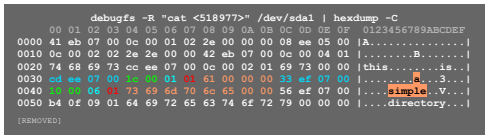
لاحظ كل دليل يجب أن يبدأ بالمدخلات "." و ".." وهي روابط تشير إلى الدليل الحالي "."والدليل الأم ".." (هذه الحالة الوحيدة التي يسمح فيه لينكس بالروابط إلى الأدلة) وكل مدخلة دليل تتضمن التالي

رقم مؤشر الفهرسة Inode number | طول مدخلة الدليل Total entry length (مدخلات الدليل ستكون متغيرة الطول بسبب الاسم. ويجب أن تكون محاذية للحدود 4 بايت. لذلك، طول "." 12 بايت حتى وإن كان 9 بايت فقط)

طول اسم الملف File name length (مثلا كان 1 بايت في مدخلة ".") | نوع الملف File type (في حالة "." و ".." سيكون "2"، ويعني دليل "directory")

اسم الملف File name (لاحظ أن بقية بايتات الإضافية في مدخلة الدليل تتضمن فقط فراغات nulls لكن الاسم لا ينتهي بـ null)

أخيرا، لحظ حقل طول المدخلة في مدخلة الملف الأخيرة في الدليل. الذي كان 0x0FB4 ، أو 4020 بايت. هذه المدخلة تستهلك بقية بايتات حتى نهاية الكتلة (4020 + 76 = 4096). لأن مدخلة الدليل الأخيرة دائما تكون محاذية للنهاية الكتلة. ومدخلات الدليل لا يمكنها أن تتجاوز حدود الكتلة. الآن راقب ماذا يحدث بعد حذف الملف "simple" من الدليل:



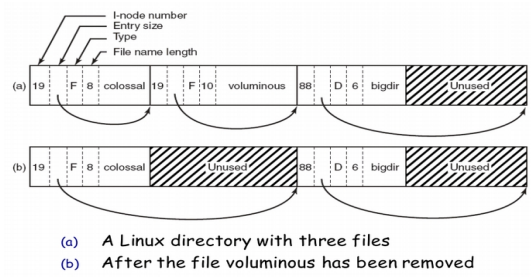
طول مدخلة الملف "a"، الذي كان سابقا 12 بايت، أصبح الآن 28 بايت (0x1C)، بعد استهلاك مدخلة الملف المحذوف "simple". لكن طول اسم الملف في مدخلة الملف "a" لا يزال 1 بايت فقط. بقية المدخلة مجرد مساحة مهملة " slack"، لكنها تحتفظ بالمدخلة القديمة للملف المحذوف. وهذا هو السلوك المعياري عند حذف الملفات في أنظمة ملفات يونكس الكلاسيكية المدخلة التي قبل الملف المحذوف ببساطة ستنمو حتى تستهلك المدخلة المحذوفة "deleted" ما عدا ذلك مدخلة الملف المحذوف ما زالت لم تتغير ويمكن استعادتها أو استخراجها can be carved for.

مثال آخر، في نظام ملفات ext2

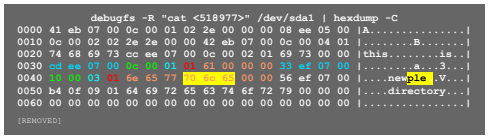
- (a) دليل لينكس يتضمن 3 ملفات
- (b) بعد حذف ملف voluminous

ترجمة هذه الحقول :

- رقم مؤشر الفهرسة I-node number
- حجم المدخلة Entry size
- نوع الملف type
- طول اسم الملف File name length
- مساحة غير مستخدمة Unused

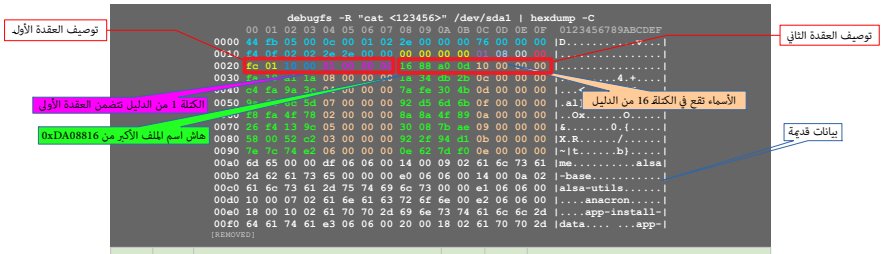


على أية حال، هذه المساحة المهملة "slack" من مدخلات الدليل المحذوفة يمكن استخدامها مرة أخرى بإضافة ملفات جديدة إلى الدليل. مثلا، لاحظ ماذا يحدث عند إضافة الملف "new" إلى نفس الدليل:



طول مدخلة ملف "a" عاد 12 بايت. ومدخلة الملف الجديد "new" بطول 16 بايت. يمكنك رؤية الثلاثة بايت الأخيرة من اسم ملف "simple" بعد "new" هذا يعني أن ext لا يهمه حشو الفراغات الإضافية في مدخلات الدليل الجديدة

الأدلة الكبيرة - إذا كانت الأدلة مجرد لوائح للملفات بدون ترتيب، فالبحث عن المدخلة يعني تحليلا متتابعا لعدد كبير من مدخلات الدليل. وبنمو حجم الدليل، متوسط زمن البحث average search time سوف يتصاعد خطيا. في معظم الأنظمة الحديثة حل هذه المشكلة كان بتنظيم مدخلات الدليل في بنية بيانات تقبل البحث. مثلا، نظام ملفات NTFS يستخدم B-trees. بينما مطوري ext3 أنشؤوا نظام شجرة الهاش hashed tree، المعروف باسم "HTree"، والذي أصبح معيار في ext4. ويمكنك رؤية تطبيق ذلك إذا تجاوز الدليل حجم الكتلة الواحدة. مثال أول كتلة من دليل /usr/share/doc :



إزاحة	حجم	حقل	إزاحة	حجم	حقل
31	1	أعلام (غير مستخدمة، عادة تكون 0)	0	4	مدخلات "." و ".."
32	2	أقصى عدد ممكن للتسجيلات dx_entry	24	4	محجوزة (صفر) Reserved / zero
34	2	العدد الفعلي للتسجيلات dx_entry التابعة	28	1	خوارزمية هاش Hashing algorithm
36	4	رقم الكتلة النسبية من أجل "zero hash"	29	1	حجم تسجيلات dx_entry (عادة يكون 8)
--	--	بقية تسجيلات dx_entry	30	1	عمق الشجرة depth of tree

أولا، لاحظ أن مدخلات "." و ".." ما زالت تظهر في بداية الكتلة، هذا للتوافق مع الإصدارات السابقة. لاحظ أيضا أن طول مدخلة ".." هو 0x0FF4 أي 4084 بايت، (التي تشير إلى نهاية الكتلة)، كي تظهر المدخلة وكأنها تستهلك بقية الكتلة (4084 = 12 + 4096) وهذه أيضا ميزة للتوافق خلفيا، لإخفاء بيانات htree عن أية شفرة قديمة تحاول ترجمة الدليل كمتتالية بسيطة من مدخلات الملفات.

بعد 24 بايت الأولى، بقية الكتلة تستخدمها بنية dx_root للتعريف بجذر htree. تقنيا، مدخلات "." و ".." جزء من dx_root وهذه البنية تستهلك الكتلة الأولى، والحقول المهمة في الكتلة تبدأ عند بايت 24ـ مع ترويسة فهرس الهاش hash index header :

بايتات 24-27 تشير إلى 4 بايت صفرية null بايت 28 يشير إلى نوع خوارزمية هاش التي تستخدمها htree، (القيمة 0x01 تبدوا معيارية في ext4، وتشير إلى خوارزمية ترتكز على MD4)

بايت 29 يشير إلى حجم تسجيلات dx_entry، المستخدمة في فهرسة مختلف الكتل في htree. (هذه التسجيلات ستكون دائما بطول 8 بايت. وسنشرح ذلك لاحقا).

بايتات 32-33 تشير إلى العدد الأقصى من تسجيلات dx_entry (أي node descriptors) التي يمكن حشوها في هذه الكتلة بعد الحقول الأولية في بنية dx_entry. (في المثال كانت 508 والفعلية 16)

تسجيلات dx_entry تبدأ عند بايت 40 داخل الكتلة، (أو تحديدا تبدأ مع توصيف العقدة الثانية التي تتضمن ملفات مع هاش اسم ملف) لذا القيمة القصوى ستكون: حجم كتلة 4096 بايت ناقص 40 بايت من حقول dx_entry، مقسومة على حجم تسجيلات dx_entry ـ(8 بايت) تساوي القيمة القصوى ((4096 – 40) ÷ 8 =507) أي إمكانية وجود 507 توصيف عقدة node descriptors في الكتلة. لكن القيمة الفعلية 0x01FC أو 508 لأن مدخلة "zero hash" في بايتات 36-39 تحسب كتسجيلة إضافية dx_entry. وتشير إلى أن الكتلة 1 من الدليل تتضمن أول عقدة first node.

وباعتبار أن كتلة dx_entry يمكنها فهرسة أكثر من 500 كتلة htree، والكتل بدورها يمكن أن تتضمن المئات من مدخلات أسماء الملفات، لذلك نادرا ما تحتاج htree لأكثر من مستوى واحد.

بايت 30 يشير إلى "عمق الشجرة" الذي دائما 0x00، للإشارة إلى أن الشجرة مسطحة flat tree. (مواصفة ext4 تسمح بوجود الشجرة المتداخلة nested tree، لكن لم أجد لها تطبيق حتى الآن).

بايتات 34-35 تتضمن العدد الفعلي للتسجيلات dx_entry التي تتبع (هنا كانت 16 فقط في الاستخدام)، مع حساب تسجيلة "zero hash" (أي first node) باعتبارها إحدى تسجيلات dx_entry.

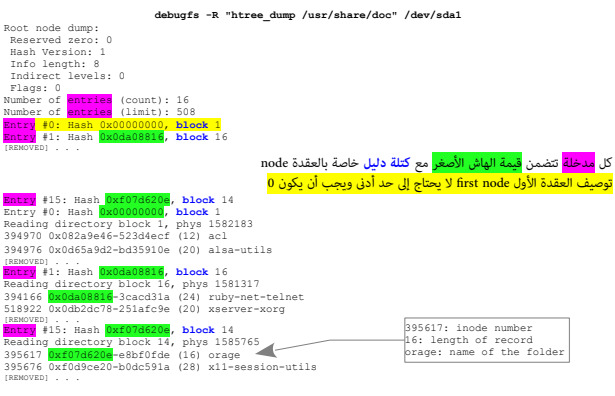
كل تسجيلة dx_entry بطول 4 بايت، قيمة هاش hash value متبوعة بإزاحة الكتلة النسبية 4 بايت relative block من بداية ملف الدليل (أي الدليل).

في المثال التالي نتفحص التسجيلات الأولى من dx_entry المجدولة كالتالي:

إزاحة الكتلة	قيمة هاش	إزاحة الكتلة	قيمة هاش	إزاحة الكتلة	قيمة هاش	إزاحة الكتلة	قيمة هاش	إزاحة الكتلة	قيمة هاش	إزاحة الكتلة	قيمة هاش	إزاحة الكتلة	قيمة هاش
1	"zero hash" 0x00000000	16	0xDA08816	8	0x1AA119FA	12	2BDB341A	4	3C9AFAC4	...	...	...	...

تسجيلات dx_entry ستكون في شكل جدول بحث lookup table ومرتبة بواسطة قيمة هاش hash value. المدخلة الابتدائية "zero hash" تعني أن جميع الملفات التي تم هاش أسمائها إلى قيم hash to values أقل من 0xDA08816 يمكن إيجادها في رقم الكتلة 1 من ملف الدليل. وأسماء الملفات التي تملك قيمة هاش أكبر من أو يساوي 0xDA08816 لكنها أقل من 0x1AA119FA يمكن إيجادها في الكتلة 16 من الملف، ... إلى آخره. تسجيلات dx_entry مرتبة بقيمة هاش hash value حتى تستطيع شفرة نظام ملفات ext عمل بحث ثنائي وإيجاد إزاحة الكتلة المناسبة بشكل أسرع.

لإيجاد الحد العلوي upper bound من قيم الهاش hashes في العقدة، نبحث في المدخلة عن العقدة التالية next node، التي كانت في المثال عند بايت 30ـ وتملك قيمة الهاش 0x1AA119FA.



بعد التسجيلة الأخيرة dx_entry، (أي last node descriptor) بقية الكتلة مساحة مهملة slack (أي مخصصة وبدون استخدام) عادة، هذه المساحة المهملة تتضمن بيانات من مدخلات دليل سابقة عندما كان الدليل يتناسب في كتلة واحدة. مثلا مباشرة بعد نهاية التسجيلة الأخيرة dx_entry يمكنك رؤية جزء من مدخلة ثم المدخلة الأصلية للدليل الفرعي"alsa-utils" (نوع الملف 02) ثم مدخلة للدليل فرعي آخر يسمى"anacron" عند مؤشر الفهرس 0x606E1.

عادة، ستجد أيضا مدخلات حية لتلك الأدلة في كتلة htree أيا كانت التي تم الهاش إليها hashed into. لكن احتمال أن هذه الأدلة تم حذفها فيما بعد وأن هذه المدخلات في مساحة الدليل المهمة قد تكون التسجيلة الوحيدة لوجودهم.

```
#debugfs -R "cat <123456>" /dev/sda1 | hexdump -C
      00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000  44 fb 05 00 0c 00 01 02  2e 00 00 00 76 00 00 00  |D...........v...|
0010  f4 0f 02 02 2e 2e 00 00  00 00 00 00 01 08 00 00  |................|
0020  fc 01 10 00 01 00 00 00  16 88 a0 0d 10 00 00 00  |................|
0030  fa 19 a1 1a 08 00 00 00  1a 34 db 2b 0c 00 00 00  |.........4.+....|
0040  c4 fa 9a 3c 04 00 00 00  7a fe 30 4b 0d 00 00 00  |...<....z.0K....|
0050  9a 61 6c 5d 07 00 00 00  92 d5 6d 6b 0f 00 00 00  |.al]......mk....|
0060  f8 fa 4f 78 02 00 00 00  8a 8a 4f 89 0a 00 00 00  |..Ox......O.....|
0070  26 f4 13 9c 05 00 00 00  30 08 7b ae 09 00 00 00  |&.......0.{.....|
0080  58 00 52 c2 03 00 00 00  92 2f 94 d1 0b 00 00 00  |X.R....../......|
0090  7e 7c 74 e2 06 00 00 00  0e 62 7d f0 0e 00 00 00  |~|t.....b}.....|
00a0  6d 65 00 00 df 06 06 00  14 00 09 02 61 6c 73 61  |me..........alsa|
00b0  2d 62 61 73 65 00 00 00  e0 06 06 00 14 00 0a 02  |-base...........|
00c0  61 6c 73 61 2d 75 74 69  6c 73 00 00 e1 06 06 00  |alsa-utils......|
00d0  13 00 07 02 61 6e 61 63  72 6f 6e 00 e2 06 06 00  |....anacron.....|
00e0  18 00 10 02 61 70 70 2d  69 6e 73 74 61 6c 6c 2d  |....app-install-|
00f0  64 61 74 61 e3 06 06 00  20 00 18 02 61 70 70 2d  |data.... ...app-|
[REMOVED]
```

بقية ملف الدليل (أي الدليل) هي كتل طرفية "leaf blocks"في شجرة htree. هذه الكتل مملوءة بمدخلات الدليل العادية (أي linked list) وتقرأ ببساطة بشكل متتابع. ولأن صيغة htree مصممة للتوافق خلفيا، الشفرات الأقدم لا يزال بإمكانها القيام بالبحث المتتابع العادي من خلال مدخلات الدليل. إن كنت تفكر في استعادة ملفات الأدلة المحذوفة، يجب أن تعلم أن ملفات الأدلة التي بحجم أكبر من كتلة واحدة ستكون مجزئة عادة. خوارزمية تخصيص الكتل في ext تضع الملفات مع الدليل الأم في نفس مجموعة الكتل. والدليل الذي ينمو مع الوقت ويتجاوز حجم الكتلة الواحدة، سيستهلك جميع الكتل المجاورة.

118. ^ كل مدخلة تتضمن قيمة الهاش hash value الأصغر وكتلة دليل للعقدة. لكن أول توصيف عقدة node descriptor لا يحتاج إلى حد أدنى minimum ويجب أن يكون 0. لذلك، تستخدم 4 بايت لغرض تخزين العدد الحالي والعدد الأقصى من توصيف العقد الذي يمكن أن يتناسب داخل الكتلة. ولهذا، أول توصيف عقدة سيملك الحقول التالية (كما تظهر في the first node descriptor) :

عنوان كتلة أول عقدة Block address of first node - العدد الحالي من توصيفات العقد Current number of node descriptors - العدد الأقصى من توصيفات العقد Maximum number of node descriptors

بقية الكتلة بعد توصيف العقدة الأخير(ة)، تتضمن بيانات من مدخلات الدليل السابقة.

119. ^ الدليل الرئيسي التالي من أحد مستخدمي لينكس، يملك تمثيل البيانات التالية على جهاز التخزين: تخطيط بيانات الدليل المتصلة، (مع حجم كتلة 4 كيلوبايت)

```
$ ls -la ~
.
..
.bash_profile
.bashrc
mbox
public_html
tmp
```

مدخلة الدليل 0

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 783362	4	0
طول التسجيلة : 12	2	4
طول الاسم : 1	1	6
نوع الملف : EXT2_FT_DIR=2	1	7
اسم : .	1	8
حشو [98]	3	9

مدخلة الدليل 1

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 1109761	4	12
طول التسجيلة : 12	2	16
طول الاسم : 2	1	18
نوع الملف : EXT2_FT_DIR=2	1	19
اسم : ..	2	20
حشو	2	22

مدخلة الدليل 2

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 783364	4	24
طول التسجيلة : 24	2	28
طول الاسم : 13	1	30
نوع الملف : EXT2_FT_REG_FILE	1	31
اسم : bash_profile.	13	32
حشو	3	45

مدخلة الدليل 3

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 783363	4	48
طول التسجيلة : 16	2	52
طول الاسم : 7	1	54
نوع الملف : EXT2_FT_REG_FILE	1	55
اسم : .bashrc	7	56
حشو	1	63

مدخلة الدليل 4

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 783377	4	64
طول التسجيلة : 12	2	68
طول الاسم : 4	1	70
نوع الملف: EXT2_FT_REG_FILE	1	71
اسم : mbox	4	72
حشو		

مدخلة الدليل 5

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 783545	4	76
طول التسجيلة : 20	2	80
طول الاسم : 11	1	82
نوع الملف :EXT2_FT_DIR=2	1	83
اسم : public_html	11	84
حشو	1	95

مدخلة الدليل 6

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 669354	4	96
طول التسجيلة : 12	2	100
طول الاسم : 3	1	102
نوع الملف :EXT2_FT_DIR=2	1	103
اسم : tmp	3	104
حشو	1	107

مدخلة الدليل 7

قيمة	حجم (بايت)	إزاحة (بايت)
رقم مؤشر الفهرسة : 0	4	108
طول التسجيلة : 3988	2	112
طول الاسم : 0	1	114
نوع الملف : EXT2_FT_UNKNOWN	1	115
اسم :	0	116
حشو	3980	116

120. ^ أ، ب، ت، ث شجرة المديات Extent Tree: ذكرنا سابقا، أن العدد الأقصى للمديات في inode هو 4. وأن 16 بت فقط تمثل عدد الكتل في المدى. في الواقع، بت العليا محجوزة (تستخدم لوسم المدى "reserved but initialized" التي هي جزء من ميزة التخصيص المسبق في ext4)، هذا يعني أن المدى يمكن أن يتضمن كحد أقصى 2^15 كتلة أي 128 ميغابايت على افتراض أن حجم الكتلة 4 كيلوبايت. ورغم أن 128 ميغابايت حجم كبير، لكن ماذا يحدث إذا كان حجم الملف أكبر من نصف جيجابايت ؟ في هذه الحالة نحتاج إلى أكثر من 4 مديات لفهرسة هذا الملف بالكامل. أيضا ماذا يحدث إذا كان الملف صغير لكن يملك تجزئة كثيرة ؟ هنا أيضا نحتاج إلى مديات أكثر في تمثيل مجموعة الكتل التي تشكل الملف. في الأمثلة التالية سنستخدم نفس الأجراء السابق، مع ملف يدعى sci-dictionary.pdf.

```
# ls -li sci-dictionary.pdf
548546 -rw-rw-r-- 1 xxx xxx 11087886 Jun 27 05:55 sci-dictionary.pdf
```

عندما يحتاج ext4 إلى استخدام أكثر من 4 مديات سوف ينشئ بنية شجرية على القرص لحفظ الحقول الضرورية للمدى. وهذا ما يخبرنا به حقل "عمق الشجرة" في ترويسة المديات.

العقد الطرفية leaf nodes في قاع الشجرة هي هياكل اعتيادية للمدى (مثل تلك التي شاهدناها في الأمثلة الأخرى). أما العقد الداخلية interior nodes في بقية الشجرة فهي بنية مختلفة تسمى "مؤشر المدى" أو "فهرس المدى" extent index. نحن نعلم أننا نتعامل مع بنية extent index لأن عمق الشجرة ليس صفر (أنظر للطرح الست عشري)، إذن نحن لسنا عند عقدة طرفية leaf node.

```
dd if=/dev/sda1 skip=2098698 bs=4096 count=506 | dd skip=6113 bs=256 count=1 | hexdump -Cv
      00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000  b4 81 e8 03 0e 30 a9 00  2e ae db 5b 2e ae db 5b  |.....0.....[...[|
0010  d3 18 33 5b 00 00 00 00  e8 03 01 00 a8 54 00 00  |..3[.........T..|
0020  00 00 08 00 01 00 00 00  0a f3 01 00 04 00 01 00  |................|
0030  00 00 00 00 00 00 00 00  5f 89 20 00 00 00 1c 00  |........_. .....|
0040  02 00 00 00 1e 00 00 00  c1 0c 14 00 20 00 00 00  |............ ...|
0050  60 00 00 00 85 0f 14 00  80 00 00 00 80 01 00 00  |`...............|
0060  b3 98 14 00 20 d6 a1 2b  00 00 00 00 00 00 00 00  |.... ..+........|
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0080  20 00 00 00 98 e4 08 6f  30 b7 3d 2a 90 24 4b 7e  | ......o0.=*.$K~|
0090  e2 17 33 5b f4 05 4f 14  00 00 00 00 00 00 00 00  |..3[..O.........|
[REMOVED]
```

تحليل حقول ترويسة المديات extent header :

2 بايت تشير إلى التوقيع 0xF30A 2 بايت تشير إلى عدد المديات في inode الذي كان 1 ، 2 بايت تشير إلى عدد المديات الأقصى في inode الذي كان 4

2 بايت تشير إلى عمق الشجرة. الذي كان هذه المرة 1، وليس 0 4 بايت تشير إلى هوية أو رقم توليد الشجرة generation ID كان أيضا صفر

تحليل حقول فهرس المدى extent index :

بايتات من 52 إلى 55 تشير إلى رقم الكتلة المنطقية (0x00000000) بايتات من 56 إلى 59 تشير إلى عنوان الكتلة الفيزيائية (32 بت المنخفضة) (0x0020895F). بايتات من 60 إلى 61 تشير إلى عنوان الكتلة الفيزيائية (16 بت العليا) (0x0000) بايتات من 62 إلى 63، غير مستخدمة.

ترويسة المديات تحتوي قيمتين أساسيتين القيمة الأولى هي الكتلة المنطقية أين نجد المديات تحت هذه العقدة في الشجرة. وستكون أول 4 بايت من بنية extent index. في هذا المثال المديات في الشجرة الثانوية تبدأ عند الكتلة المنطقية صفر. هذا يعني بداية الملف. القيمة الأخرى في extent index هي رقم كتلة (البيانات) الفيزيائية التي تتضمن معلومات عن المستوى التالي في الشجرة. ومثل بقية عناوين الكتل في ext4، قيمة 48-بت ستكون من مجزئين : 32 بت منخفضة و 16 بت عليا. في هذا المثال كانت 2132319 = 0x0000020895F، التي بالأحرى تعني إزاحة الكتلة.

بقية 16 بت في extent index شاغرة. ربما كنت تتوقع أن تكون صفر. لكنها بالقيمة 0x001C. في الواقع، رغم أن بنية ترويسة المديات في inode تشير إلى استخدام فقط البنية الأولى extent index، هناك حقول أخرى للبنية المدى تظهر أيضا في بايتات من 64 إلى 99 ليست صفر، لكن لماذا ؟ سوف نعرف الجواب فيما بعد، الآن دعنا نتفحص الكتلة 2132319.

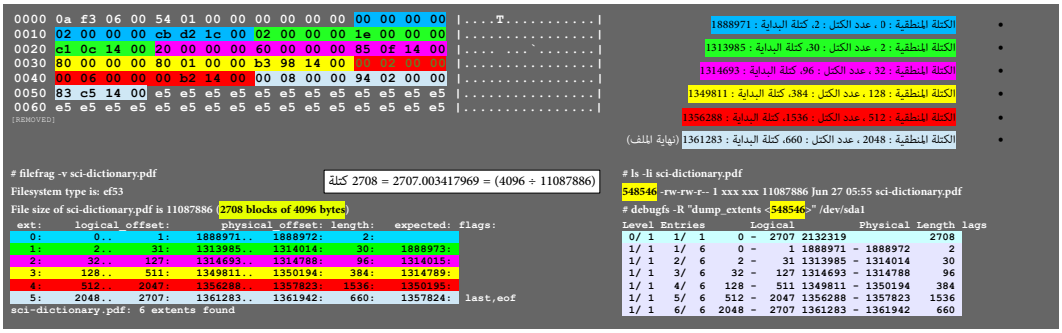
كل كتلة بيانات تستخدم في تخزين معلومات شجرة المديات ستبدأ ببنية ترويسة مديات خاصة، تماما مثل تلك الموجودة أعلاه في inode.

في الطرح التالي تظهر أول 112 بايت من كتلة 2132319، هنا حقول ترويسة مديات ملونة:

```
blkcat /dev/sda1 2132319 | hexdump -C
      00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000  0a f3 06 00 54 01 00 00  00 00 00 00 00 00 00 00  |....T...........|
0010  02 00 00 00 cb d2 1c 00  02 00 00 00 1e 00 00 00  |................|
0020  c1 0c 14 00 20 00 00 00  60 00 00 00 85 0f 14 00  |.... ...`.......|
0030  80 00 00 00 80 01 00 00  b3 98 14 00 00 02 00 00  |................|
0040  00 06 00 00 00 b2 14 00  08 00 00 94 02 00 00     |...............|
[REMOVED]
```

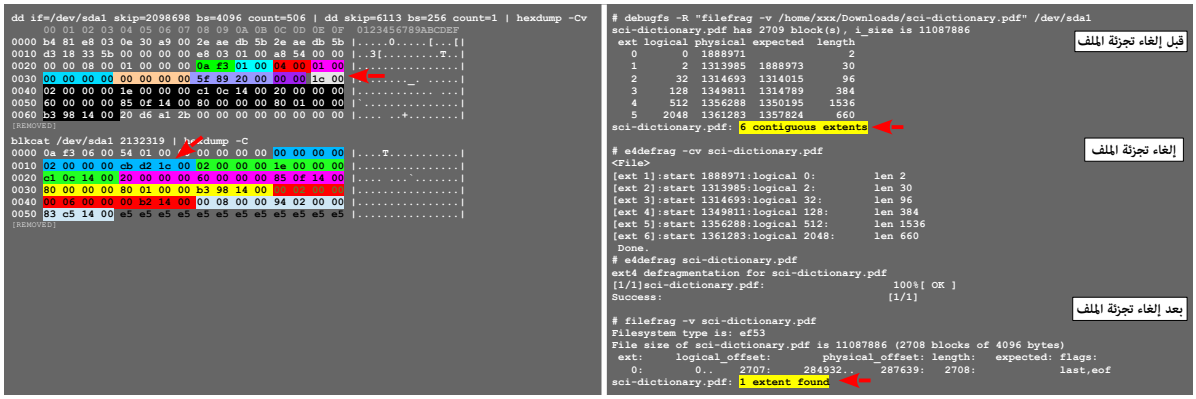
2 بايت تشير إلى التوقيع 0xF30A بايتات من 6 إلى 7 تشير إلى عمق الشجرة. الذي كان صفر. (0x0000) إذن نحن نزلنا الآن إلى مستوى واحد في الشجرة وأية هياكل مديات نجدها بعد الترويسة ستكون مديات اعتيادية وليست هياكل extent index. بايتات من 2 إلى 3 تشير فعليا إلى 6 مديات لكن قيمة حقل عدد المديات الأقصى كانت 340 = 0x0154.

تذكر أننا نستخدم كتلة بيانات كاملة من 4096 بايت. ترويسة المديات عند بداية الكتلة تستهلك منها 12 بايت، وتبقى 4084 بايت لحفظ هياكل المديات. يبدوا أن هياكل المديات (340 × 12) تحتل 4080 بايت من البيانات، إذن ذلك يعني أقصى ما يمكننا حشوه في المساحة الشاغرة في هذه الكتلة. أيضا من غير المحتمل في العمليات العادية أن تحتاج تجزئة الملف إلى أكثر من 340 مدى في تمثيل الملف. مع الحجم الأقصى 128 ميغابايت لكل مدى. 340 مدى يمكنها عنونة ملفات تصل إلى 42,5 جيجابايت. أكبر من ذلك سنحتاج إلى زيادة في حجم شجرة المديات. دعنا الآن نحاول فك شفرة هذه المديات 6 :



عقد داخلية Interior nodes في شجرة المديات عقد طرفية leaf nodes في شجرة المديات. تنبيه: طول وسلسلة الكتل الخاصة بالمدى الأخير في العقدة الداخلية هو تخمين من دوال مكتبة المديات extents library functions، وليس مخزن في هياكل بيانات نظام الملفات. ومن ثم، القيم التي تظهر ليست بالضرورة دقيقة، كما أنها لا تشير إلى مشكلة أو فاسد في نظام الملفات.

ذكرنا سابقا أن هياكل المدى الشاغرة في inode كانت تملك بيانات في داخلها. إذا تفحصت ذلك جيدا، سترى أن البيانات في هياكل مديات inode في بايتات من 64 إلى 99 التي عادة تحتفظ بالمديات من 2 إلى 4 تتطابق تماما مع البيانات في المديات من 2 إلى 4 في الكتلة 2132319. ذكرنا أيضا أن 2 بايت العليا من بنية extent index في inode التي عادة شاغرة، كانت تملك بعض البيانات في داخلها. أيضا إذا قارنت ذلك سترى أن بايتات "1C 00" في 2 بايت الأخيرة من بنية extent index تتطابق مع 2 بايت الأخيرة من بنية المدى الأول #1 في الكتلة 2132319. إذن ما الذي يحدث هنا ؟. يبدو أن شفرة ext4 بطيئة lazy بعض الشيء (راجع تأثير delayed allocation). وهذا الملف. إما أنه مستمر في النمو و/ أو مستمر في التجزؤ، fragmenting وبالتالي نظام الملفات يستمر في إضافة المديات في Inode. عندما يحتاج إلى مدى خامس، يرفع قيمة حقل عمق الشجرة في ترويسة المديات ويستبدل أول مدى بالبنية extent index. والشفرة لا تهتم بإعادة كتابة المديات الشاغرة في inodes ولا تهتم بتصفير 2 بايت الشاغرة الأخيرة في بنية extent index.



لفهم عمل آلية إلغاء تجزئة الملف راجع كتيب "Outline of Ext4 File System & Ext4 Online Defragmentation Foresight"

121. ^ أ، ب، ت، ث، ج، الخصائص الممتدة EA وأذون قوائم التحكم بالنفاذ acl الممررة كخصائص ممتدة بين النواة ومساحة المستخدم، هي من الوسائل الفعالة والبسيطة في حماية أنظمة لينكس الداخلية، إلى جانب التطبيقات الأخرى مثل SELinux. وللتعامل مع الخصائص الممتدة، المستخدم في حاجة إلى فهم استخدام أداوت chattr و lsattr، الأولى تقوم بتعيين وإزالة الخصائص الممتدة (باستثناء بعض الخصائص)، والثانية (مثل ls) تقوم بعرض الخصائص المرتبطة بالملف. هناك الكثير من الخصائص الممتدة التي يمكنك إضافتها أو إزالتها من الملفات، مثلا: لجعل الملف ثابت، غير قابل للتغيير. حتى من قبل المستخدم الجذر، نستخدم الخاصية "i" مثال: chattr +i some_special_file. بالمناسبة الأمر ls لن يعرض هذه الخاصية الموجودة في الملف، فقط أداة lsattr تستطيع عرضها. ولإزالة أية خاصية ممتدة من الملف، نستخدم خيار "S" مثال: chattr -S some_file. أمثلة في التطبيق:

جعل الدليل /data غير قابل للتغيير: immutable (I) حتى المستخدم الجذر لا يستطيع حذف الدليل

إضافة خاصية إلحاق البيانات append only إلى نهاية الملف حيث لا يمكن الكتابة فوق البيانات الموجودة. هذا مفيد في إعادة توجيه خرج إلى ملفات log

لاحظ في نظام الملفات، جميع الملفات والأدلة ستملك مسبقا خاصية المديات extent format (e)

```
$ chattr +i data/
$ chattr +a log.txt
$ lsattr
----i---------e---- ./data
-------------e---- ./textfile.txt
-------------e---- ./file1.txt
----a---------e---- ./log.txt
$ rm -rf data
rm: cannot remove 'data': Operation not permitted
$ echo "text" > log.txt
-su: log.txt: Operation not permitted  (لا يمكن الكتابة فوق البيانات القديمة داخل الملف)
$ echo "text" >> log.txt  (يمكن فقط إلحاق البيانات بالملف)
```

مبدئيا، فقط المستخدم الجذر يستطيع تغيير الخصائص الممتدة. إذا أراد المدير السماح للمستخدمين تعيين أو إزالة هذه الخصائص الممتدة، يجب وصل نظام الملفات مع خيار user_xattr مثال:

```
UUID=661ab9f1-c381-4962-bcfc-0b5e2aab1ce9 /home ext4 defaults,user_xattr,acl   1 2  (ملف etc/fstab)
```

لكن استخدام الخيارات المبدئية للوصل بدلا من استخدام مدخلة /etc/fstab مفيد مع الأجهزة الأقراص الخارجية. مبدئيا هذه الخيارات ستكون موجودة، عند إنشاء نظم الملفات ext2/3/4 مثال:

```
$ cat /etc/mke2fs.conf | grep acl
default_mntopts = acl,user_xattr
```

حينذاك يمكنك تعيين وعرض الخصائص الممتدة بواسطة setfattr و getfattr. مثال:

```
$ setfattr -n user.comment -v "this is a comment" testfile
$ getfattr testfile
# file: testfile
user.comment
$ getfattr -n user.comment testfile
# file: testfile
user.comment="this is a comment"
```

عند تعيين أذون ACLs على الملفات، نستخدم setfacl و getfacl. عموما حتى تستطيع استخدام هذه الأدوات مع ACLs، يجب وصل نظام الملفات عن طريق خيار acl :

```
$ tune2fs -l /dev/sda1 | grep "Default mount options:"  (تأكد من وجود الخيار)
Default mount options:    user_xattr acl
$ tune2fs -o acl /dev/sda1  (يمكن acl إذا كان غير موجود)
```

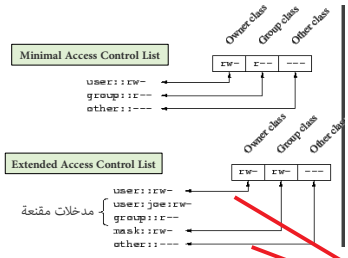
تقليديا، الملفات تملك ثلاثة تصاريح أو أذون مختلفة هي: قراءة، وكتابة، وتنفيذ (rwx)، من أجل ثلاثة أصناف أو مجموعات مختلفة هي: المستخدم user، والمجموعة group، والآخرون other. يمكن للمدير إعطاء أشخاص معينين صلاحية الكتابة إلى ملف معين، عبر إنشاء مجموعة تملك تلك الصلاحية ثم ضم أولئك الأشخاص إلى المجموعة. لكن مع ACLs أنت تتجاوز الحاجة إلى إنشاء مجموعة.

مثلا، لنفترض أن هناك ملف باسم testfile مملوك من قبل Mohammed:Mohammed مع أذون 0644 (هنا محمد يملك حق القراءة والكتابة، وغيره يملك حق القراءة فقط)، إذن فقط محمد يستطيع تعديل ذلك الملف. إذا أردنا أن يملك مثلا السيد علي ali حق الكتابة إلى ذلك الملف، هنا نستخدم ACLs:

```
$ setfacl -m u:ali:rw testfile
$ getfacl testfile
# file: testfile
# owner: Mohammed
# group: Mohammed
user::rw-
user:ali:rw-
group::r--
mask::rw-
other::r--
% ls -al testfile
-rw-rw-r--+ 1 Mohammed   Mohammed   6 2009-11-11 14:28 testfile
```

في هذا المثال أعلاه، تم تعديل أذون ACLs في الملف testfile بإضافة ACL من أجل السيد ali الذي أصبح يملك إذن القراءة والكتابة (rw). باستخدام getfacl، يتضح لنا أن ali يملك نفس الأذون rw، مثل Mohammed. باستخدام أداة ls. يمكن رؤية أن الملف يملك أيضا خاصية ACL لوجود علامة + في سلسلة الخصائص والأذون. ورغم أن الملف مملوك من طرف Mohammed (المستخدم والمجموعة) الآن ali أيضا يستطيع تعديل الملف testfile، في ألأنويه الحديثة هذه الخيارات ستعمل على أي نظام ملفات، شرط وصل نظام الملفات باستخدام acl المناسب مع خيارات user_xattr.

للمزيد من المعلومات، راجع استخدام الأدوات getfacl, setfacl, chattr, lsattr, getfattr, setfattr و acl. ووثائق توزيعات لينكس مثل archlinux 1، 2 و ubuntu. و POSIX ACLs.



خرج أداة getfacl / هذا المثال من صفحة man

```
1:  # file: somedir/
2:  # owner: lisa
3:  # group: staff
4:  user::rwx
5:  user:joe:rwx               #effective:r-x
6:  group::rwx                 #effective:r-x
7:  group:cool:r-x
8:  mask:r-x
9:  other:r-x
10:  default:user::rwx
11:  default:user:joe:rwx       #effective:r-x
12:  default:group::r-x
13:  default:mask:r-x
14:  default:other:---
```

الأسطر 4، 6، و 9 نقترن بحقول user، و group، و other في بتات أذون أنماط الملفات. هذه الثلاثة تسمى مدخلات دنيا أساسية (Minimal Access Control List) في ACL.

الأسطر 5 و 7 تشير إلى مدخلات المستخدم والمجموعة المسماة named user، named group. السطر 8 يشير إلى قناع الحقوق النافذة effective rights mask. هذه المدخلة تحد من الحقوق النافذة الممنوحة إلى جميع المستخدمون والمجموعات المسماة. (أذون الآخرون others ومالك الملف owner لا تتأثر بقناع الحقوق النافذة، لكن جميع المدخلات الأخرى تتأثر) الأسطر من 10 إلى 14 تعرض الأذون المبدئية default ACL المرتبطة بهذا الدليل. الأدلة يمكنها أن تملك أذون default ACL. والملفات الاعتيادية Regular files لا يمكنها أبدا أن تملك أذون default ACL

صيغة	نوع مدخلة ACL	
user::rwx	مالك	Owner
user:name:rwx	مستخدم مسمى	Named user
group::rwx	مجموعة مالكة	Owning group
group:name:rwx	مجموعة مسماة	Named group
mask::rwx	قناع	Mask
other::rwx	آخرون	Others

مثال آخر: المستخدم john يملك الأذون النافذة r-x : لكنه لا يملك حق الكتابة إلى الملف، لأن w لم يعين في القناع.

أذون	نوع
user:john:rwx	مستخدم مسمى
mask::r-x	قناع

122. ^ أ، ب، ت، ربط البيانات تعني عملية إنشاء روابط بين العناصر المختلفة في أنماط أو نماذج البيانات. ويستخدم ربط البيانات كخطوة أولية في مهام تكامل البيانات المتنوعة؛ والتي تشمل:

- تحويل البيانات أو وساطة البيانات بين مصدر البيانات والهدف/المقصد.
- تعريف العلاقات بين البيانات كجزء من تحليل نسب البيانات data lineage analysis.
- اكتشاف البيانات الحساسة مثل آخر أربعة أرقام من رقم الضمان الاجتماعي المخفي في معرف مستخدم آخر كنوع من تقنيع (إخفاء) البيانات أو مشروع إلغاء أو إعادة التعريف.
- دمج قواعد بيانات متعددة في قاعدة بيانات واحدة وتعريف أعمدة البيانات المكررة لدمجها أو إزالتها.

123. ^ أ، ب، ت، flush مسح أو حذف شيء زائد عن الحاجة، أو إجهاض عملية. مثال: "All that nonsense has been flushed." في يونكس، تعني إجبار كتابة بيانات وحدات الإدخال والأخراج في الصون buffered I/O إلى القرص،كما يفعل نداء fflush هذا ليس إجهاض أو حذف كما في المعنى الأول، بل هو طلب إفراغ مبكر.مثل تخليص بيانات عناوين الكتل في الصوان بكتابتها إلى القرص.

تخليص الخابية cache flushing : عندما تصل كمية البيانات الغير مكتوبة في خابية إلى مستوى معين، المتحكم يقوم دوريا بكتابة البيانات التي في الخابية إلى القرص. هذه الكتابة تدعى flushing البيانات التي تم قبولها في write cache جهاز القرص ستكتب أخيرا إلى أطباق القرص. على شرط أن لا تحدث حالة starvation نتيجة خلل في البرنامج الثابت، أو انقطاع مصدر طاقة القرص قبل إجبار كتابات الخابية إلى أطباق القرص. وللتحكم في write cache مواصفة ATA تتضمن الأوامر FLUSH CACHE (E7h) و FLUSH CACHE EXT (EAh) التي ستجعل القرص يكمل كتابة البيانات من الخابية. وسيعود القرص في حالة جيدة بعد كتابة بيانات write cache إلى القرص. أيضا يمكن استهلال flushing على الأقل في بعض الأقراص عبر استبداء لين Soft reset أو أمر وضع استعداد Standby.

تخليص الخابية cache flushing مستخدم بشكل إجباري في لينكس في تطبيق حواجز الكتابة write barriers في أنظمة الملفات مثل ext4، مع أمر الكتابة FUA لكتل تنفيذ قيد الحوادث. [؟؟]

124. ^ في الحاسوب، الذاكرة الوسيطة للقرص أو صِوان القرص، disk buffer (أحيانا تسمى: ذاكرة القرص المخبئية disk cache، أو صِوان مَخبأ cache buffer) هي ذاكرة مضمنة في القرص الثابت، تعمل كذاكرة وسيطة بين بقية الحاسوب وطبق القرص الفيزيائي المستخدم في التخزين. حجم الذاكرة الوسيطة في الأقراص الثابتة الحديثة من 8 إلى 256 ميغابايت، وفي SSD ، تصل إلى 1 جيجابايت.

125. ^ صوان أو ذاكرة وسيطة أو مَحفظ Buffer هو مكان مؤقت في ذاكرة تخزين فيزيائية / ملموسة حيث يتم فيه تخزين البيانات حين تنقّل من مكان إلى آخر. ويتعلق تركيبه (خصائصه) حسب الوظيفة التي يشغلها. عادة البيانات تخزن في Buffer بالشكل الذي تأتى منه كأي وحدة إدخال (مثل لوحة المفاتيح) أو قبل أن ترسل إلى أي وحدة إخراج (مثل الطابعة) ويمكن أستخدامها أثناء نقل البيانات بين العمليات التي تجرى داخل الحاسوب. وهناك خوارزميات مختلفة وكثيرة تستخدم لأجل Buffers مثل (FIFO أو shelf) و(LIFO أو stack) و double buffering و Circular buffer.

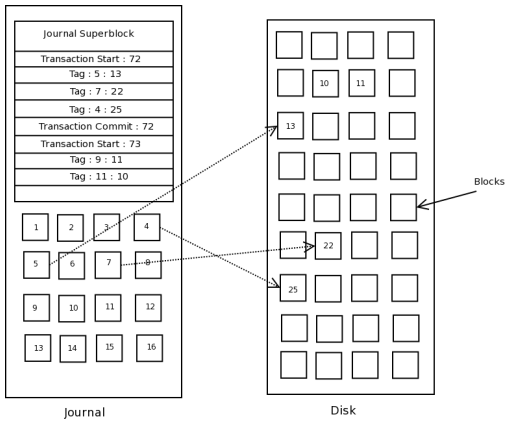
126. ^ تخصيص متعدد للكتل Multi-Block Allocation. عملية تخزين مجموعة من البيانات سويةً عوضًا عن تخزينها واحدة واحدة مما يقلل استهلاك الموارد ويزيد السرعة.

127. ^ مثال على الأخطاء في نظام الملفات والتي لا تعني بالضرورة وجود مشكلة في نظام الملفات!:

```
Filesystem state:         clean with errors
FS Error count:           2188
First error time:         Sun Nov 29 08:13:23 2015
First error function:     ext4_find_entry
First error line #:       1309
First error inode #:      2
First error block #:      0
Last error time:          Thu Dec 24 06:49:42 2015
Last error function:      ext4_find_extent
Last error line #:        901
Last error inode #:       113836936
```

128. ^ أ، ب أنواع البيانات التي تحمل إشارة تحفظ القيم الموجبة والسالبة والتي لا تحمل إشارة تحفظ القيم الموجبة الكبيرة ولا تحفظ السالبة وهناك طرق لتمثيل القيم التي تحمل إشارة، منها المتمم الثنائي

129. ^ المدخلات الأحادية في descriptor block تسمى entries أو descriptor tags (الترجمة قد تعني لواحق كتلة التوصيف!) كهياكل صغرى journal descriptor block تخزن تعين كل صِوان بيانات وصفية في journal إلى موقعها الفعلي على القرص في شكل tags، بعد ذلك يتم تخليص metadata buffers إلى journal. وعند امتلاء descriptor باللواحق tags أو عند تخليص جميع metadata buffers إلى journal، يتم كذلك تخليص journal descriptor إلى journal. الآن أصبحت جميع metadata buffers في journal ومواقعها الفعلية على القرص مسجلة. هذه البيانات الحاضرة، يمكن استعمالها لاستعادة النظام.



130. ^ أدلة يونكس / لينكس

شرح	رمز	اسم الدليل / الحاوية	
أعلى دليل (عقدة) في أية بنية ملفات في يونكس	/	دليل جذر	Root Directory
دليل (ملفات) المستخدم !	~	دليل منزل	Home Directory
الموقع الاعتيادي عند العمل مع الملفات (Working directory)	.	دليل حالي	Current Directory
دليل مباشرة فوق الدليل الحالي	..	دليل أم	Parent Directory
المسار: لائحة من الأسماء مفصولة بالشرطة "/". المسار المطلق Absolute Path : يتتبع المسار من الجذر إلى الملف أو الدليل، ودائما يبدأ بالدليل الجذر (/) مثال /home/student/Desktop/assign1.txt. والمسار النسبي Relative Path : يتتبع المسار من الدليل الحالي، ولا يستهل بالشرطة المائلة "/". . مثال: Desktop/assign1.txt			

131. ^ TSK أداوت للتحليل الجنائية، تعمل من سطر أوامر يونكس، في تفحص أنظمة الملفات ووحدات التخزين، وتدعم أنظمة الملفات كثيرة (مثل ext4 منذ 4.1.0) الواجهة الرسومية لإداوات هي Autopsy.

في TSK اسم الأداة مركب من جزأين، الأول للتعريف بالمجموعة والثاني بالوظيفة، مثلا fls تعني أداة في فئة أسماء الملفات (f) مع وظيفة السرد (ls)، الأدوات مرتبة في مجموعات وطبقات، تقريبا كالتالي:

أدوات القرص	Disk Tools	فئة البيانات الوصفية	Metadata Category
أدوات نظام وحدة التخزين	Volume System Tools	فئة أسماء الملفات	File Name Category
أدوات نظام الملفات	File System Tools	فئة التطبيقات	Application Category
فئة نظام الملفات	File System Category	فئة متعددة	Multiple Category
فئة المحتوى	Content Category	أداوت للبحث	Searching Tools

أداة TSK	وظيفة	
blkcat (1) (سابقا dcat)	Views the contents of a block / Display the contents of file system data unit in a disk image.	عرض محتويات كتلة (عرض وحدة بيانات Data unit * نظام الملفات في صورة القرص)
blkls (1) (سابقا dls)	List or output file system data units	سرد أو خرج (طرح) وحدات بيانات نظام الملفات (أي الكتل)
blkcalc (1) (سابقا dcalc)	Converts between unallocated disk unit numbers and regular disk unit numbers	التحويل بين أرقام وحدات القرص التي بدون تخصيص وأرقام وحدات القرص الاعتيادية
blkstat (1) (سابقا dstat)	Display details of a file system data unit (i.e. block or sector)	عرض تفاصيل عن وحدة بيانات نظام الملفات (أي، الكتلة أو القطاع)
fcat (1)	Output the contents of a file based on its name	خرج محتويات الملف يرتكز على لسم الملف
ffind (1)	Finds the name of the file or directory using a given inode	إيجاد اسم الملف أو الدليل الذي يستخدم مؤشر الفهرسة المحدد
fls (1)	List file and directory names in a disk image	سرد أسماء الملفات والأدلة في صورة القرص
fsstat (1)	Display general details of a file system	عرض تفاصيل شاملة عن نظام الملفات
ils (1)	List inode information	سرد معلومات مؤشر الفهرسة (سرد محتويات مديات الملف على القرص).
istat (1)	Display details of a meta-data structure (i.e. inode)	عرض تفاصيل بنية البيانات الوصفية (أي، مؤشر الفهرسة)
icat (1)	Output the contents of a file based on its inode number	خرج محتويات الملف يرتكز على رقم مؤشر فهرسة الملف
hfind (1)	Lookup a hash value in a hash database	البحث عن قيمة الهاش في قاعدة بيانات الهاش
ifind (1)	Find the meta-data structure that has allocated a given disk unit or file name	إيجاد بنية البيانات الوصفية التي قامت بتخصيص وحدة القرص أو اسم الملف المحدد
jcat (1)	Show the contents of a block in the file system journal	إظهار محتويات كتلة في (ملف) قيد حوادث نظام الملفات
jls (1)	List the contents of a file system journal	سرد محتويات قيد حوادث نظام الملفات
usnjls (1)	List the contents of a NTFS Update Sequence Number journal	سرد محتويات USN Journal في نظام ملفات NTFS
mmcat (1)	Output the contents of a partition to stdout	خرج (أو طرح) محتويات القسم على stdout (خرْج قياسي)
mmls (1)	Display the partition layout of a volume system (partition tables)	عرض تخطيط الأقسام في نظام وحدة التخزين (جداول الأقسام)
mmstat (1)	Display details about the volume system (partition tables)	عرض تفاصيل عن نظام وحدة التخزين (جداول الأقسام)
tsk_comparedir (1)	compare the contents of a directory with the contents of an image or local device	مقارنة محتويات الدليل مع محتويات الصورة أو الجهاز المحلي
tsk_gettimes (1)	Collect MAC times from a disk image into a body file	جمع MAC times من صورة قرص في ملف متن (شفرة)
tsk_loaddb (1)	populate a SQLite database with metadata from a disk image	تزويد قاعدة بيانات SQLite بالبيانات الوصفية من صورة القرص
Tsk_recover (1)	Export files from an image into a local directory	تصدير الملفات من الصوة إلى الدليل المحلي
sigfind (1)	Find a binary signature in a file	إيجاد التوقيع الثنائي في الملف
srch_strings (1)	Display printable strings in files	عرض السلاسل الصالحة للطباعة في الملف
img_cat (1)	Output contents of an image file	خرج محتويات ملف صورة
img_stat (1)	Display details of an image file	عرض تفاصيل ملف صورة
jpeg_extract (1)	Jpeg extractor is used by fiwalk, provided by TSK to manipulate JPG files	jpeg_extractor تستخدمها fiwalk (في معالجة ملفات JPG)
fiwalk (1)	print the filesystem statistics and exit	طباعة إحصائيات نظام الملفات ثم الخروج
sorter (1)	Sort files in an image into categories based on file type	فرز ملفات في الصورة في فئات (تصنيفات) على أساس نوع الملف
mactime (1)	Create an ASCII time line of file activity	إنشاء مخطط زمني بشفرة أسكي للنشاط الملف

(* Data unit = a grouping of consecutive sectors)

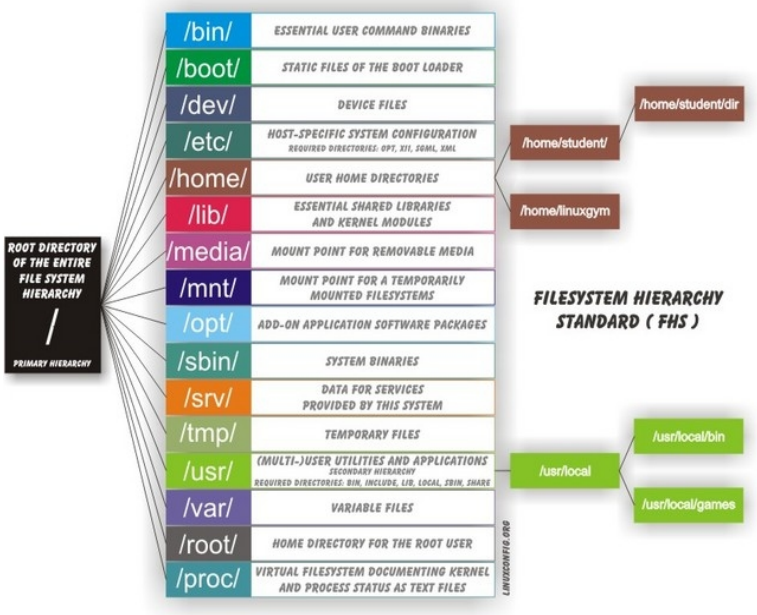
132. ^ أ، ب، ، مؤشر الفهرسة المعزول orphaned inode هو ذلك inode الذي ليس له رابط (مدخلة دليل) لكن لا يزال مفتوح من عملية أخرى process مثلا، عند تنفيذ tail -f myfile متبوع بتنفيذ rm myfile على نفس الملف من صدفة أخرى. نظام الملفات يتعقب orphaned inodes كي يستطيع تنظيفها فيما بعد عند خروج/توقف العملية، (أداة fsck تنشئ لذلك مدخلات دليل جديدة في lost+found). الملفات المحذوفة يمكن عرضها بأداة lsof. وأرقامها بواسطة:

```
# fsstat /dev/sdXY | grep -i orphan
```

133. ^ جذور نظام الملفات

- اندرو ستيوارت تانينباوم كتب مينيكس وأصدره في 1987 لغرض تدريس تصميم أنظمة التشغيل وعمل النواة (على IBM PC/AT)، ثم تمّ طرحه كنظام مفتوح المصدر في 2000 تحت رخصة BSD
- نظام ملفات مينيكس أول نظام ملفات في لينكس، يملك بنية تحاكي UFS من 6 عناصر area, data area, inodes, zone bitmap, inode bitmap, superblock, Boot sector.
- لينوس تورفالدس استعمل نظام ملفات مينيكس أثناء كتابة أول نسخة من نواة لينكس (1991).
- نظام الملفات الممتد ext كان أول تطبيق يستخدم VFS و أول نظام ملفات أُنشئ خصيصًا لنظام التشغيل لينكس، وأطلق في أبريل 1992.
- رمي كارد صمم ext لتخطى حدود نظام ملفات مينيكس. ويغلب على ext بنية البيانات الوصفية المستوحاة من نظام ملفات يونكس FFS/UFS.
- رمي كارد صمم أيضا ext2 المستخدم في نواة لينكس كبديل لنظام ext.
- ext3 المزود بنظام قيد حوادث journaling من تطوير ستيفن تويدي. استعمل بكثرة في توزيعات جنو لينكس GNU Linux كنظام أساسي لإدارة الملفات على الأقراص.
- ext4 يملك ميزّات إضافية ومن تطوير: Mingming Cao, Andreas Dilger, Alex Zhuravlev (Tomas), Dave Kleikamp, Theodore Ts'o, Eric Sandeen, Sam Naghshineh

LINUX MINIX IBM PC



التسلسل الهرمي القياسي لنظام الملفات

134. ^ حواجز الكتابة Write Barriers :

آلية في النواة (في نظام الإدخال/الإخراج الكتلي الفرعي) تستخدم للضمان تنظيم ordered وكتابة written البيانات الوصفية للنظام الملفات على جهاز التخزين المستقر persistent storage (الغير متطاير) حتى في حالة انقطاع الكهرباء عن أجهزة التخزين ذات خابية الكتابة المتطايرة volatile write caches. في حال تمكين Write Barriers أنظمة الملفات تتأكد أيضا من استقرار البيانات المرسلة عبر()fsync أثناء انقطاع الكهرباء. لكن تمكين حواجز الكتابة Write Barriers يأثر سلبا في أداء بعض التطبيقات، والتطبيقات التي تستخدم دائما ()fsync أوتعمل على إنشاء وحذف الملفات الصغيرة الكثيرة، ستعمل أبطأ. في الأساس ما يسمى الحاجز barrier يمنع كتابة أية كتل بعد barrier إلى أن يتم كتابة جميع الكتل قبل تنفيذ barrier إلى الوسيط.

135. ^ نواة لينكس تدعم أنظمة الملفات (الفيزيائية) التالية، ReiserFS، UFS، UDF، OCFS، NSS، MINIX fs، JFS، JFFS2، JFFS، ISO 9660، HPFS، HFS، FAT، F2FS، ext4، ext3، ext2، cramfs، BFS، Amiga FFS

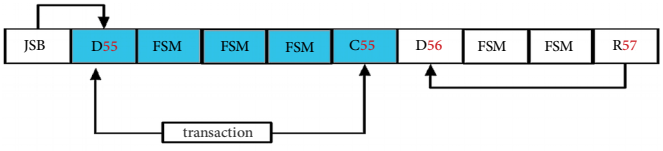
QNX4 FS و Acorn ADFS (دعم الكتابة تجريبي) و HFS+ (دعم الكتابة محدود/فقط مع قيد الحوادث الخالي)، و System V FS (دعم الكتابة؟) و BeFS، EFS، FreeVxFS و Reiser4، Btrfs، Tux3، ZFS (دعم الكتابة تجريبي وحاليا معطوب) و NTFS (دعم كتابة/قراءة مع مشغل مساحة المستخدم إضافي) و exFAT (يتطلب مشغل مساحة المستخدم) exFAT fs على FUSE. أيضا نواة لينكس تدعم أنظمة الملفات الشبكية التالية: NFS، AFS، CIFS، Coda، 9P، Ceph.

136. ^ آلية journal تقتفي أثر التغيرات (updates) في نظام الملفات (إنشاء/حذف/تعديل) باستخدام متتابعات من الإجراءات transactions (لخفض عمليات disk I/O). كل متتابعة الإجراء tx تملك رقم #tx وتتكون من هذه العناصر:

- كتلة توصيف Descriptor block: كل إجراء tx يبدأ بكتلة تصف بداية transaction تصف الكتل اللاحقة في journaled، تشمل المواقع النهائية للكتل على القرص.

  في نمط data هذه الكتلة تتبعها كتل البيانات الوصفية وكتل البيانات، بينما في نمط ordered أو write back تتبعها فقط كتل البيانات الوصفية. (في خرج JLS كتلة نظام الملفات المقابلة ستظهر بشكل # FS Block)

  - كتلة بيانات وصفية Metadata block: هناك كتلة بيانات وصفية واحدة أو أكثر لكل transaction، هذه الكتل تخزن التغييرات.
- كتلة تنفيذ Commit block: وفقا للنمط journal mode، هذه الكتلة تشير إلى نهاية إجراء ناجح transaction قبل حفظ نقطة الفحص! checkpoint. (مزامنة log مع نظام الملفات)
- كتلة إلغاء Revoke block: تحتفظ بلائحة من كتل نظام الملفات لن تطبق أثناء الاسترداد. بسبب خطأ أثناء العملية أو حذف ملف وإعادة استخدام inode، الذي يلغى من log عبر هذا transaction.



JSB	Journal Super Block	الكتلة العليا ضمن ملف قيد الحوادث (v1 أو v2)
D[X]	Descriptor Block X	كتلة التوصيف (رقم)
FSM	File System Metadata	بيانات وصفية من نظام الملفات
C[X]	Commit Block X	كتلة تنفيذ (رقم)
R[X]	Revoke Block X	كتلة إلغاء (رقم)

في هذا الشكل، يظهر الإجراء 55 transaction مكتمل، لأن مجموعة كتل البيانات الوصفية المحاصبة للإجراء تنطوي على descriptor block و commit block. أما الإجراء 56 transaction فغير مكتمل لذلك لم تنشأ commit block. عند وصل نظام الملفات مرة أخرى وفحص journal، سيتم إنشاء revoke block وبما أن 56 transaction لا يملك commit block ويملك sequence number أقل من 57، التغييرات المصاحبة لهذا الإجراء لن يتم تكرارها replayed.

137. ^ تحليل ext3 journal (مثال من كتاب Brian Carrier) :

```
# icat -f linux-ext3 dev/hdb2 8 | xxd
      0001 0203 0405 0607 0809 0A0B 0C0D 0E0F 0123456789ABCDEF
0000 c03b 3998 0000 0004 0000 0000 0000 0400 .;9.............
0016 0000 0400 0000 0001 0000 0126 0000 0000 ...........&....
0032 0000 0000 0000 0000 0000 0000 0000 0000 ................
0048 a36c 4be5 c222 460b b76f d45b 518b 083c .lK.."F..o.[Q..<
0064 0000 0000 0000 0000 0000 0000 0000 0000 ................
0080 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

بايتات من 0 إلى 3 تشير إلى التوقيع magic number بايتات من 4 إلى 7 تشير إلى نوع الكتلة الذي كان 4 ويشير إلى استخدام Journal superblock, v2 بايتات من 8 إلى 11 تشير إلى رقم المتتالية الذي كان 0 (transaction ID) بايتات من 12 إلى 15 تشير إلى أن حجم كتلة journal كان 1,024 بايت بايتات من 16 إلى 19 تشير إلى وجود 1,024 كتلة في هذا journal بايتات من 20 إلى 23 تشير إلى بداية مدخلات/لواحق journal في الكتلة 1 من journal لمعرفة أول إجراء transaction في journal، ننظر في بايتات من 24 إلى 27 التي تشير إلى أول رقم متتالية 294 (0x0126) بايتات من 28 إلى 31 تشير إلى وجود الأجزاء في الكتلة 0.

رأينا أعلاه أن لواحق journal تبدأ عند الكتلة 1. هنا سبب وجود أول إجراء في الكتلة 0 هو فصل نظام الملفات بشكل نظيف واكتمال جميع transactions (الصفر في هذا الحقل لا يعني أن journal نظيف!).

بعد وصل نظام الملفات وإنشاء ملف في الدليل الجذر root directory. الآن كتلة superblock تتضمن التالي (الطرح بناء على رقم journal inode) :

```
# icat -f linux-ext3 dev/hdb2 8 | xxd
      0001 0203 0405 0607 0809 0A0B 0C0D 0E0F 0123456789ABCDEF
0000 c03b 3998 0000 0004 0000 0000 0000 0400 .;9.............
0016 0000 0400 0000 0001 0000 0127 0000 0001 ...........'....
```

بسبب وجود إجراءات صالحة valid transactions في ملف journal (أي تملك نهاية إجراء) الآن رقم المتتالية في البداية ارتفاع إلى 295 (0x0127) مع تعيين كتلة journal المقابلة إلى الكتلة 1.

سنتفحص مضمون الكتلة 1 في journal (تذكر أن هذا رقم كتلة في journal وليس في نظام ملفات). يمكننا رؤية هذه الكتلة بتمرير خرج icat مع dd باستخدام حجم الكتلة 1024 بايت، أو نستخدم jcat:

```
# jcat -f linux-ext3 dev/hdb2 1 | xxd
      0001 0203 0405 0607 0809 0A0B 0C0D 0E0F 0123456789ABCDEF
0000 c03b 3998 0000 0001 0000 0127 0000 0004 .;9........'....
0016 0000 0000 0000 0000 0000 0000 0000 0000 ................
0032 0000 0000 0000 0002 0000 0002 0000 000e ................
0048 0000 0002 0000 0005 0000 0002 0000 00a3 ................
0064 0000 0002 0000 0003 0000 000a 0000 0000 ................
0080 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

بايتات من 4 إلى 7 تشير إلى نوع الكتلة 1 أي descriptor block رقم متتالية الكتلة كان 295 (0x0127) أول لاحقة توصيف Descriptor tag تبدأ عند بايت 12، وكانت للكتلة 4 في نظام الملفات

بايتات من 16 إلى 19 تشير إلى حقل الأعلام الذي كان 0، ويعني أن حقل UUID موجود في 16 بايت التالية.

هذه اللاحقة (tag) تشير إلى أن الكتلة التي تتبع كتلة descriptor تقابل الكتلة 4 في نظام الملفات. وهذه تستخدم من أجل inode bitmap، وتم تحديثها updated عند تخصيص inode جديد

اللاحقة/المدخلة الثانية تبدأ عند بايت 36، بايتات من 36 إلى 39 تشير إلى إنها من أجل الكتلة 2 في نظام الملفات قيم الأعلام كانت 2، و تعني أن حقل UUID غير موجود.

هذه اللاحقة تشير إلى أن الكتلة الثانية بعد كتلة descriptor كانت من أجل الكتلة 2 في نظام الملفات، وهذه تستخدم من أجل جدول group descriptor table.

عند تحليل بقية الكتلة، يمكننا رؤية أن: الكتلة 14 تم تحديثها لأنها في جدول مؤشرات الفهرسة inode table وتتضمن inode المخصص للملف الجديد؛ الكتلة 5 تم تحديثها لأنها تتضمن inode الخاص بالدليل الجذر الكتلة 163 تم تحديثها لأنها مكان تخزين مدخلات الدليل الجذر directory entries. الكتلة 3 تم تحديثها لأنها تتضمن المصفوفة الثنائية للكتل block bitmap.

ملاحظة: كانت ستكون هناك لاحقة/مدخلة أخرى من أجل مضمون الملف الجديد. لو أننا استخدمنا نمط data=journal، في journal الذي يسجل أيضا تحديثات المضمون content updates،

هناك 6 لواحق/مدخلات في جدول التوصيف descriptor، لذلك سنتفحص الكتلة 8 في ملف journal من أجل كتلة التنفيذ commit block.

```
# jcat -f linux-ext3 dev/hdb2 8 | xxd
      0001 0203 0405 0607 0809 0A0B 0C0D 0E0F 0123456789ABCDEF
0000 c03b 3998 0000 0002 0000 0127 0000 0000 .;9........'....
0016 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

بايتات من 4 إلى 7 تشير إلى أنها كتلة تنفيذ (0x02) بايتات من 8 إلى 11 تشير إلى أنها من أجل المتتالية 295 (0x127).

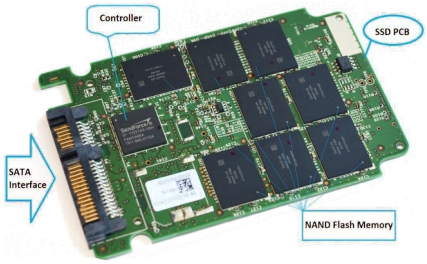
كمثال أخير, كتلة descriptor تشير إلى أن الكتلة 6 في ملف journal تتضمن كتلة نظام ملفات للدليل الجذر root directory. عند طرح مضمون الكتلة 6 يمكننا رؤية تجديد الكتلة بمدخلة new-file.txt

```
# jcat -f linux-ext3 dev/hdb2 6 | xxd
      0001 0203 0405 0607 0809 0A0B 0C0D 0E0F 0123456789ABCDEF
0000 0200 0000 0c00 0100 2e00 0000 0200 0000 ................
0016 0c00 0200 2e2e 0001 0b00 0000 e803 0c00 ................
0032 6e65 772d 6669 6c65 2e74 7874 0c00 0000 new-file.txt....
```

هذا يسمح للمستخدم تحديد الملفات التي تم مؤخرا إنشائها وحذفها من النظام. أداة jls في TSK ستعرض مضمون journal. وهذا خرج من المثال السابق:

```
# jcat -f linux-ext3 dev/hdb2
JBLK    Description
0:      Superblock
1:       Allocated Descriptor Block  (seq: 295)
2:       Allocated FS Block 4
3:       Allocated FS Block 2
4:       Allocated FS Block 14
5:       Allocated FS Block 5
6:       Allocated FS Block 163
7:       Allocated FS Block 3
8:       Allocated Commit Block  (seq:  295)
9:       Unallocated FS Block Unknown
[removed]
```

138. ^ أ، ب، SSD لا تملك أجزاء متحركة تتعطل ميكانيكيا. لكنها تملك كتل (كتابة ومسح البيانات إلكترونيا) كل كتلة في SSD ترتكز على الذاكرة الوميضية flash-based يمكن مسحها (وإعادة كتابتها) مرات محدودة قبل أن تخسر. المتحكمات تحاول إدارة هذا الحد (عبر تقنية Wear leveling التي تحاول توزيع الكتابة بالتساوي على خلايا flash كي تظل الأقراص أطول مدة في الاستخدام العادي.SSD التي ترتكز على ذاكرة الوصول العشوائي الديناميكية DRAM ليس لها حد في عدد مرات الكتابة، لكن تصبح غير صالحة للاستعمال عند فشل المتحكم controller. أيضا SSDs كثيرا ما تخسر بسبب التيار الكهربائي. و Reliability تختلف حسب الصانع والطراز. هناك عدة أشياء يمكنها أن تأثر على فترة حياة lifespan قرص SSD مثل: درجة الحرارة temperature ووضع السبات hibernation الذي يحفظ الكثير من محتويات RAM على القرص، واستخدام SSD في بيئة مركز بيانات حيث يمكن للكتابات الكثيرة إلى سجلات log files أن تأثر على فترة حياة SSD (لذلك البعض يقترح وضع دليل var/log على قرص persistent ramdisk). على أية حال، الخطوات التالية ستخفض من دورات الكتابة-وإعادة الكتابة (الغير ضرورية) write-rewrite cycles على كتل SSD :



- تعيين علم noatime الذي يعطل atime في inodes (الملفات والأدلة) (راجع أداة MOUNT(8) و fstab)

```
/etc/fstab
UUID=41b6b187-be76-4447-b18b-d39cc744b184 /      ext4   noatime,discard,data=writeback,errors=remount-ro 0 1
tmpfs                                     /tmp   tmpfs  nodev,nosuid,noexec,mode=1777                  0 0
```

- تمكين ميزة discard والتوقف عن إلغاء تجزئة الأقراص Defragging (تنبيه: تأكد من دعم TRIM في SSD قبل تمكين الميزة: hdparm -I /dev/sda | grep TRIM)

لتمكين الميزة sdXY/dev/ sdXY -o discard /dev/ tune2fs (راجع أيضا الفرق بين Periodic TRIM و Continuous TRIM و Trim an entire device على مواقع الويكي archlinux و wikipedia)

- إنشاء journal على HDD منفصل، أيضا تعطيل journal قد يحسن من الأداء لكن عند انقطاع الكهرباء أو الجهد الحاد spikes أو إقفال للنواة kernel lockup المستخدم سيخسر البيانات، خصوصا في غياب دعم البطّاريّة battery-backed cache، أو وجود UPS أو حتى فقط الحماية من تقلّبات التيار Surge protector.
- إنشاء قسم أو منطقة swap area على قرص HDD بدلا من قرص SSD. على أية حال، في لينكس إصدارة Ubuntu الأخيرة، تقوم آليا (زمن التنصيب) بإنشاء ملف swapfile في الدليل الجذر.
- إن كنت تملك ما يكفي من RAM: يمكنك وصل الدليل /tmp على tmpfs (وليس Ramfs الذي ينمو ديناميكياً) بإضافة ما يشبه هذا السطر في ملف fstab : `tmpfs /tmp tmpfs nodev,nosuid,noatime,size=2G,mode=1777 0 0`
- في حال كان RAM أكبر من 4GB ضع Firefox cache في دليل /tmp (في RAM): قد تحتاج إلى إنشاء مفتاح browser.cache.disk.parent_directory في about:config وتعيينه إلى /tmp وتأكد من ذلك في about:cache.
- إذا كنت تملك قرص SSD واحد فقط ولا تستخدم التخزين الخارجي كثيرا، يمكنك إضافة elevator=noop إلى معاملات إقلاع النواة في ملف /etc/default/grub في سطر GRUB_CMDLINE_LINUX_DEFAULT، حتى وإن كنت تستخدم أحيانا USB sticks. أو في حالة وجود عدة أقراص، بعد تنصيب حزمة sysfsutils أضف السطر التالي block/sda/queue/scheduler = noop إلى ملف /etc/sysfs.conf.
- أيضا هناك من يوصي باستخدام stride و stripe-width على أقراص SSD مع ext4 !
- أخيرا، لا تنسى دائما عمل نسخ احتياطي دوري للملفات المهمة (على وسيط خارجي).

139. ^ نواة لينكس 4.1 تدعم التشفير في ext4 على مستوى الأدلة والملفات الأحادية. عند تشفير دليل فارغ. أية ملفات تنشأ في الدليل ستكون مشفرة أيضا. لكن فقط أسماء الملفات والمحتوى سيكون مشفر، البيانات الوصفية لا تشفر، و inline data ليست مدعومة في الملفات. ولأن تشفير مضمون الملف يتم مباشرة في الذاكرة، التشفير سيكون متوفر فقط إذا كان حجم العنقود بحجم PAGE_SIZE أي يساوي 4k.

140. ^ معظم توزيعات لينكس لا تدعم هذه الميزة؛ لا قبل و لا بعد إنشاء نظام الملفات مثال على ذلك e2fsprogs 1.44.1 في Ubuntu. في بعض الأنظمة يمكن تمكين الميزة عبر e2fsprogs 1.43 لكن برنامج GRUB (2.02) لا يدعم الميزة وستكون هناك مشاكل في الإقلاع في حال تمكين الميزة مثلا عبر debugfs. أداة lsattr يمكنها عرض علم الخاصية inline data (N) الذي يشير إلى وجود بيانات مضمنة في الملف لكن لا يمكن تعيين/إعادة تعيين هذا العلم باستخدام chattr. أيضا هناك معلومة تقول عند تمديد مساحة extra isize لا ينبغي تحريك system.data xattr خارج متن inode؛ تطبيق inline data يفترض عدم وجود system.data xattr في كتلة خصائص ممتدة خارجية. راجع علة البرنامج 200005

141. ^ مثال على التغيرات التي يمكن أن تحدث في Journal عند إنشاء وإعادة تسمية وحذف الملف (نظام ملفات ext3 journal).

عند إنشاء الملف textfile.txt

```
Create File - System Changes
5: 66  -> Inode Bitmap
6: 1   -> Group Descriptor Table
7: 67  -> Inode Table
8: 577 -> Data Block
9: 65  -> Data Bitmap

JBlk    Description
0:  Superblock (seq: 0)
1:  Allocated Descriptor Block (seq: 2)
2:  Allocated FS Block 67
3:  Allocated Commit Block (seq: 2)
4:  Allocated Descriptor Block  (seq: 3)
5:  Allocated FS Block 66
6:  Allocated FS Block 1
7:  Allocated FS Block 67
8:  Allocated FS Block 577
9:  Allocated FS Block 65
10: Allocated Commit Block (seq: 3)
11: Unallocated FS Block Unknown
12: Unallocated FS Block Unknown
13: Unallocated FS Block Unknown
14: Unallocated FS Block Unknown
15: Unallocated FS Block Unknown

BLOCK GROUP INFORMATION
-----------------------------------------
Number of Block Group: 8
Inodes per group: 8160
Blocks per group: 32768

Group: 0:
  Inode Range: 1 - 8160
  Block Range: 0 - 32767
  Layout:
    Super Block: 0 - 0
    Group Descriptor Table: 1 - 1
    Data bitmap: 65 - 65
    Inode bitmap: 66 - 66
    Inode Table: 67 - 576
    Data Blocks: 577 - 32767
  Free Inodes: 8148 (99%)
  Free Blocks: 32184 (98%)
  Total Directories: 2
```

Journal Block 8: FS Data Block 577

```
00008000 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 ................
00008010 0c 00 02 02 2e 2e 00 00 0b 00 00 00 14 00 0a 02 ................
00008020 6c 6f 73 74 2b 66 6f 75 6e 64 00 00 0c 00 00 00 lost+found......
00008030 d4 0f 0c 01 74 65 73 74 66 69 6c 65 2e 74 78 74 Ô...testfile.txt
00008040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00008050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00008060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

عند إعادة تسمية الملف textfile.txt

```
Rename File - System Changes
12: 67   -> Inode Table
13: 577  -> Data Block

JBlk    Description
0:  Superblock (seq: 0)
1:  Allocated Descriptor Block (seq: 2)
2:  Allocated FS Block 67
3:  Allocated Commit Block (seq: 2)
4:  Allocated Descriptor Block (seq: 3)
5:  Allocated FS Block 66
6:  Allocated FS Block 1
7:  Allocated FS Block 67
8:  Allocated FS Block 577
9:  Allocated FS Block 65
10: Allocated Commit Block (seq: 3)
11: Allocated Descriptor Block (seq: 4)
12: Allocated FS Block 67
13: Allocated FS Block 577
14: Allocated Commit Block (seq: 4)
15: Unallocated FS Block Unknown
16: Unallocated FS Block Unknown
17: Unallocated FS Block Unknown

BLOCK GROUP INFORMATION
-----------------------------------------
Number of Block Group: 8
Inodes per group: 8160
Blocks per group: 32768

Group: 0:
  Inode Range: 1 - 8160
  Block Range: 0 - 32767
  Layout:
    Super Block: 0 - 0
    Group Descriptor Table: 1 - 1
    Data bitmap: 65 - 65
    Inode bitmap: 66 - 66
    Inode Table: 67 - 576
    Data Blocks: 577 - 32767
  Free Inodes: 8148 (99%)
  Free Blocks: 32184 (98%)
  Total Directories: 2
```

Journal Block 13: FS Data Block 577

```
0000d000 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 ................
0000d010 0c 00 02 02 2e 2e 00 00 0b 00 00 00 28 00 0a 02 ............(...
0000d020 6c 6f 73 74 2b 66 6f 75 6e 64 00 00 0c 00 00 00 lost+found......
0000d030 14 00 0c 01 74 65 73 74 66 69 6c 65 2e 74 78 74 ....testfile.txt
0000d040 0c 00 00 00 c0 0f 0e 01 72 65 6e 61 6d 65 66 69 ....À...renamefi
0000d050 6c 65 2e 74 78 74 00 00 00 00 00 00 00 00 00 00 le.txt..........
0000d060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

عند حذف الملف renamefile.txt

```
Delete File - System Changes
16:577  -> Data Block
17:67   -> Inode Table
18:0    -> Super Block
19:65   -> Data Bitmap
20:1    -> Group Descriptor Table
21:66   -> Inode Bitmap

4:  Allocated Descriptor Block (seq: 3)
5:  Allocated FS Block 66
6:  Allocated FS Block 1
7:  Allocated FS Block 67
8:  Allocated FS Block 577
9:  Allocated FS Block 65
10: Allocated Commit Block (seq: 3)
11: Allocated Descriptor Block (seq: 4)
12: Allocated FS Block 67
13: Allocated FS Block 577
14: Allocated Commit Block (seq: 4)
15: Allocated Descriptor Block (seq: 5)
16: Allocated FS Block 577
17: Allocated FS Block 67
18: Allocated FS Block 0
19: Allocated FS Block 65
20: Allocated FS Block 1
21: Allocated FS Block 66
22: Allocated Commit Block (seq: 5)
23: Unallocated FS Block Unknown

BLOCK GROUP INFORMATION
-----------------------------------------
Number of Block Group: 8
Inodes per group: 8160
Blocks per group: 32768

Group: 0:
  Inode Range: 1 - 8160
  Block Range: 0 - 32767
  Layout:
    Super Block: 0 - 0
    Group Descriptor Table: 1 - 1
    Data bitmap: 65 - 65
    Inode bitmap: 66 - 66
    Inode Table: 67 - 576
    Data Blocks: 577 - 32767
  Free Inodes: 8149 (99%)
  Free Blocks: 32185 (98%)
  Total Directories: 2
```

Journal Block 16: FS Data Block 577

```
00010000 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 ................
00010010 0c 00 02 02 2e 2e 00 00 0b 00 00 00 e8 0f 0a 02 ............è...
00010020 6c 6f 73 74 2b 66 6f 75 6e 64 00 00 0c 00 00 00 lost+found......
00010030 14 00 0c 01 74 65 73 74 66 69 6c 65 2e 74 78 74 ....testfile.txt
00010040 0c 00 00 00 c0 0f 0e 01 72 65 6e 61 6d 65 66 69 ....À...renamefi
00010050 6c 65 2e 74 78 74 00 00 00 00 00 00 00 00 00 00 le.txt..........
```

مراجع

1. ^ خوارزميات وهياكل بيانات ext4 (سابقا تخطيط قرص Ext4) (المرجع الأول لهذه الكتيب / المسودة 2)
2. ^ نظام الملفات الممتدة ext4 – الموسوعة الحرة
3. ^ نظام ملفات ext2 في wiki.osdev.org
4. ^ توثيق المشروع، صفحة Ext4 Howto
5. ^ صفحة الأسئلة مكررة Frequently Asked Questions
6. ^ ميزّات ext4 الجديدة
7. ^ نظام ملفات ext4، موقع kernelnewbies.org
8. ^ ميزّات وخيارات نظام ملفات ext4 من موقع man7.org
9. ^ تخصيص متأخر للكتل Delayed allocation وصفحة الموسوعة الحرة
10. ^ دعم الحصص النسبية للقرص Design For 1st Class Quota in Ext4
11. ^ دعم تدقيق مجاميع البانات الوصفية Design for Metadata Checksums
12. ^ موضوع "Design for Large Allocation Blocks"
13. ^ موضوع "Persistant Ext4 error" و "Forking ext4 filesystem and JBD2" في lkml.org
14. ^ تخطيط / تصميم ext4
15. ^ المقارنة بين أنظمة الملفات – الموسوعة الحرة
16. ^ بعض المقالات من موقع LWN.net
17. ^ ملف ext4.txt موقع elixir.bootlin.com
18. ^ ملف ext4.h من access.redhat.com
19. ^ نظام الملفات EXT4، موقع computer-forensics.sans.org
20. ^ خصائص وأذون acl و ext3 و ext4 و fsck، من موقع archlinux
21. ^ مواضيع من موقع Linux.org و wiki.ubuntu.com
22. ^ موقع linuxconfig.org
23. ^ المقارنة بين أنوية أنظمة التشغيل
24. ^ محاولة استعادة الدليل المحذوف في ext4 ، موقع opensuse
25. ^ موضوع Ext4 on SSD Intel X25-M من marc.info
26. ^ موضوع Reserved GDT blocks من redhat.com
27. ^ مقالات عن ext4 من computer-forensics.sans.org
28. ^ معلومات من stackoverflow.com و lists.openwall.net و hackingexposedcomputerforensicsblog.blogpost.com
29. ^ تحليل بنية نظام ملفات ext4 من موقع programering.com
30. ^ موضوع Undeletion
31. ^ موضوع Ext2-Ext3-Ext4 Attributes و Ext2/Ext3/Ext4 File System من موقع softpanorama.org
32. ^ next3-utils من أمير (المشروع متوقف منذ أكتوبر 2013 ؟) Amir's ext4 snapshot work، رقع EXT4 snapshot patches في ithub.com و article.gmane.org
33. ^ موضوع "A Minimum Complete Tutorial of Linux ext4 File System" من مدونة metebalci موقع medium.com
34. ^ أذون و أنماط الملفات : أنواع الملفات في يونكس، معيار يوزيكس (POSIX)، أنماط يونكس (أذون نظام الملفات)، سي لينكس SELinux، قوائم التحكم بالنفاذ، الخصائص الممتدة للملفات، بت التقيد !، أدوات ls و Chmod ، بروتوكول تشارك الملفات NFS (نظام ملفات شبكي)
35. ^ موقع Flylib.com

    File System Forensic Analysis
    ISBN: 0321268172. EAN: 2147483647
    Year: 2006 Pages: 184. Authors: Brian Carrier
36. ^ بعض البحوث والمنشورات (أغلبها قديمة!) من مواقع بعض الجامعات الغربية. بالإضافة إلى مواقع حرة أخرى
37. ^ قاموس عرب آيز وقاموس المعاني

تنبيه

لا توجد أية مصادر عربية في هذه الكتيبات ! باستثناء بعض المصطلحات القليلة من قاموس عرب آيز وقاموس المعاني مع بعض الفقرات النادرة جدا من الموسوعة الحرة – العربية

بعض المصطلحات الواردة في هذا الكتيب، تستطيع القول أنها "صواب يحتمل الخطأ" أي ليست موجودة في أي قاموس؟ والله أعلم.

مثال على ذلك: الكتلة العليا superblock، المصفوفة الثنائية bitmap، تخليص! Flush، بت تقييد sticky bit.، إلغاء تخصيص deallocate، قيمة حارسة sentinel value، أذون نافذة/فعالة effective permissions.صور زمنية انتقائية snapshot.، التخصيص عند التخليص!- Allocate-on-flush... إلى آخره

احتمال وجود أخطاء في هذا الكتيب وارد. ووسواء كان الخطأ من المصدر الانجليزي أو من الترجمة العربية. إذا كنت متخصص أو مدون يمكنك مراجعة ومقارنة الكتيب بالمصدر الانجليزي للترجمة.

وتصحيحها في كتابتكم مع الإشارة إلى المصدر أو تصحيحها وإرسالها بالبريد الإلكتروني أو على المدونة

جهاد

جدد في محرم 1441/ سبتمبر/أيلول 2019 – تمت بحمد الله