نظام الملفات الممتد/المحسن

Extended File System

Ext4 Disk Layout

تخطيط نظام الملفات على القرص

مسوّدة

جمادى الأول / يناير/كانون الثّاني/ 2019

يُوزَّع مجّانًا ولا يُباع

محتويات

Layout

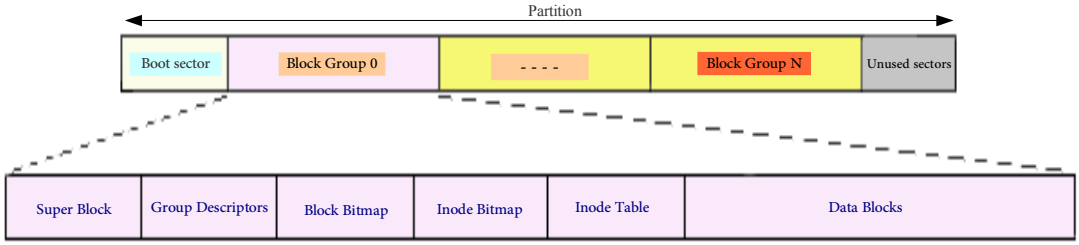
تخطيط

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 المبدئي هو 4 كيلوبايت كل مجموعة ستتضمن 32،768 كتلة (8 × 4096)، بطول 128 ميغابايت (32768 × 4096). عدد مجموعات الكتل سيكون حاصل قسمة حجم الجهاز على حجم مجموعة الكتل، 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 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 مجارية لإصدارات لينكس 4.12 Linux وحزمة 1.43-e2fsprogs.. [103]



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

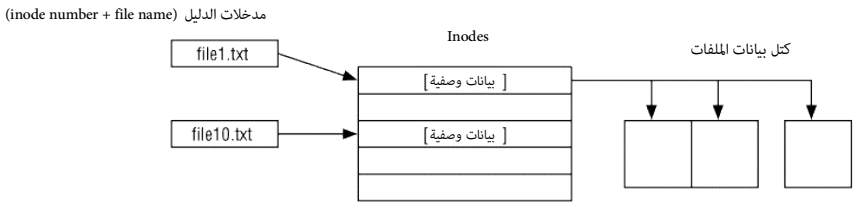
حشو في / قبل بداية المجموعة 0	الكتلة العليا	توصيفات المجموعات	كتل محجوزة للتوصيف المجموعات	مصفوفة ثنائية لكتل البيانات	مصفوفة ثنائية لمؤشرات الفهرسة	جدول مؤشرات الفهرسة	كتل بيانات
Group 0 Padding	Super Block	Group Descriptors	Reserved GDT Blocks [50]	Block Bitmap	Inode Bitmap	Inode Table	Data Blocks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

أول 1024 بايت في أو قبل مجموعة الكتل 0، محجوزة للسماح بتثبيت برامج وشفرات إقلاع x86، ولهذا كتلة superblock تبدأ عند إزاحة البايت 1024 (أيا كان حجم الكتلة)، لكن إذا كان حجم الكتلة 1024 بايت، superblock ستكون في الكتلة 1. حشوة 1024 بايت (باستثناء مجموعة الكتل 0) لا توجد في مجموعات الكتل الأخرى.

لاحظ 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 توصيف المجموعات Group Descriptors حقول grp.bg_inode_table_* ستحدد 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 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[120] extent tree، والخصائص الممتدة [121] extended attributes..

البيانات الوصفية لكل ملف و دليل (مجلد) تخزن في سجل ملف يدعى Inode (مؤشر فهرسة) [115][116][01] الذي يملك حجم ثابت ويقع في inode table. وهناك جدول واحد في كل مجموعة كتل.اسم الملف سيخزن في مدخلة الدليل [117] التي تقع في الكتل المخصصة للدليل الأم الخاص بالملف. ومدخلات الدليل عبارة عن هياكل بيانات 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، 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 4 كيلوبايت) 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						نمط	
عنصر		1 كيلوبايت	2 كيلوبايت	4 كيلوبايت	64 كيلوبايت	32-بت	64-بت
كتل	Blocks	2^{32}	2^{32}	2^{32}	2^{32}	√	
		2^{64}	2^{64}	2^{64}	2^{64}		√
مؤشرات فهرسة (سجل ملف، عقدة ملف!)	Inodes	2^{32}	2^{32}	2^{32}	2^{32}	√	√
حجم نظام الملفات	File System Size	4 تيرابايت	8 تيرابايت	16 تيرابايت	256 بيتابايت	√	√
		16 زيتابايت	32 زيتابايت	64 زيتابايت	1 يوتابايت		√
عدد الكتل لكل مجموعة	Blocks Per Block Group	8,192	16,384	32,768	524,288	√	√
عدد مؤشرات الفهرسة لكل مجموعة	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 لا تستخدم المديات (أي ربط الكتل) يجب أن توضع ضمن أول 2^{32} كتلة.والتي تستخدم المديات توضع ضمن 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 متجاورة (مبدئيا ستكون 16) 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 inode tables و bitmaps من جميع مجموعات الكتل الأخرى في flex_bg. في المثال التالي كل flex_bg بحجم 16 مع 120 مجموعة كتل (أي 8 مجموعات مرنة) [55]:

	120 Block Groups inside Flexible Block Groups															
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. مجموعة الكتل 0 تتضمن البيانات الوصفية GDT/GDG، Primary superblock, Group descriptors، Block bitmap، Inode bitmap، Inode table, Data blocks. وكما تلاحظ نسخ Superblock و group descriptors ستكون فقط في المجموعات 1، 3، 5، 7، 9، 25، 27، 49، 81،

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

الجمع بين ميزة flex_bg و ميزة sparse_super، ينتج عنه مجموعات كتل معظمها كتل بيانات فقط بدون أية بيانات وصفية metadata. 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 تحتوي: Primary superblock ، و group descriptor table، و GDT و bitmaps، و inode tables. وبسبب ميزة sparse_super، المجموعة 1 block group ستحتوي superblock، و group descriptor table.

ومن ثم واعتمادا على حجم مجموعات الكتل المرنة flex_bg ، المجموعة 16 block group يمكن أن تبدأ مجموعة كتل مرنة جديدة (أنظر للخطاطة أعلاه)، تحتوي bitmaps و inode table. بقية مجموعات الكتل في المجموعة المرنة ستكون كتل بيانات فقط، باستثناء مجموعات الكتل التي تتضمن نسخ من superblock و group descriptor table بسبب ميزة sparse_super.

مجموعات الكتل الوصفية ! Meta Block Groups

بدون الخيار META_BG[95] لاعتبارات أمنية، يتم حفظ جميع نسخ توصيف مجموعات كتل group descriptors في أول مجموعة كتل. وبالنظر إلى حجم مجموعة الكتل المبدئي 128 ميغابايت (2^{27} بايت) وتوصيف 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 $2^{27} \times 2^{21} = 2^{48}$ 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. 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 توصيف المجموعات group descriptors من مجموعة الكتل الأولى المكتظة لكامل نظام الملفات إلى أول مجموعة كتل من كل مجموعة كتل وصفية نفسها. والنسخ الاحتياطية ستكون في المجموعة الثانية والأخيرة من كل مجموعة كتل وصفية. هذا يرفع الحد الأقصى للمجموعات الكتل إلى 2^{32}، الذي يسمح بدعم نظام ملفات 512 بيتابايت.

وفي 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 تجمع لمجموعات الكتل التي يمكن وصفها بواسطة كتلة واحدة group descriptor. وبما أن حجم بنية group descriptor هو 32 بايت، مجموعة meta-block 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.

راجع الملاحظة الهامة حول BLOCK_UNINIT في فقرة المصفوفات الثنائية للمؤشرات الفهرسة وكتل البيانات block and inode bitmaps.. بيانات الجدول التالي من ملف e2fsprogs-1.44.0/tests/m_meta_bg..

	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 الميزتان meta_bg و sparse_super

كل صف يعتبر مجموعة كتل وصفية Meta Block Group، وكل خلية تعتبر مجموعة كتل. مجموعة الكتل 0 تتضمن Primary superblock (في الكتلة 1) Group descriptor, Block bitmap, Inode bitmap, Inode table, data blocks، نسخ من Superblock و group descriptors ستكون فقط في المجموعات 1، 3، 5، 7، 9، 25، 27، 49، 81، 125.

مجموعة الكتل الأولى والثانية والأخيرة في كل مجموعة كتل وصفية تبدأ مع Group descriptor إلى جانب Block bitmap, Inode bitmap, Inode table, data blocks، بقية المجموعات كتل بيانات تتضمن أيضا Block bitmap, Inode bitmap, Inode table.

التهيئة المؤجلة لمجموعة الكتل ! Lazy Block Group Initialization

الأعلام الثالثة في واصف مجموعة الكتل هي إحدى الميزّات 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 المصفوفات الثنائية inode and block bitmaps لتلك المجموعة يمكن حسابها وبالتالي كتل bitmap على القرص لن يتم تهيئها. (أي لا تصفر)، عموما هذا هو حال مجموعة الكتل الشاغرة empty block group أو مجموعة الكتل التي تتضمن فقط بيانات وصفية لمجموعة كتل ذات الموقع الثابت fixed-location block group metadata.

علم INODE_ZEROED يعني أن جدول 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 جدول 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 الخاصة التي تظهر في الجدول التالي:

مؤشر الفهرسة	ثابت / معامل	غرض
0		مؤشر الفهرسة 0 (هذا لا يوجد أصلا / لا يستخدم) [39]
1	EXT4_BAD_INO	مؤشر فهرسة قائمة الكتل المعسة (كتل تالفة)
2	EXT4_ROOT_INO	مؤشر فهرسة الدليل الجذر
3	EXT4_USR_QUOTA_INO	مؤشر فهرسة حصة المستخدم (الحصص النسبة للقرص) [13]
4	EXT4_GRP_QUOTA_INO	مؤشر فهرسة حصة المجموعة
5	EXT4_BOOT_LOADER_INO	مؤشر فهرسة محمل الإقلاع (شفرة الإقلاع)
6	EXT4_UNDEL_DIR_INO	مؤشر فهرسة دليل استرجاع الملفات المحذوفة
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]
9	EXT4_EXCLUDE_INO	مؤشر فهرسة "exclude"، لاستثناء الصور الزمنية الانتقائية snapshot (صورة تجميد أو استنسخ زمني للحالة الملفات / النظام) (؟)
10	EXT4_REPLICA_INO	مؤشر فهرسة النسخ المتماثلة، يستخدم مع بعض ميزات الغير رسمية ؟
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 mballoc[126]. 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[125][26] 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 128 ميغابايت؛ هذه الحاويات الصغرى 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 المستوي 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 الأداء 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 إلغاء 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 مجموع 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	حجم / نوع		بيانات وصفية
كامل كتلة superblock حتى حقل تدقيق المجموع. معرف UUID سيكون في داخل كتلة superblock	4	__le32	كتلة عليا
UUID + كامل كتلة MMP حتى حقل تدقيق المجموع	4	__le32	حماية من تعدد الوصل
UUID + كامل كتلة الخصائص الممتدة (حقل تدقيق المجموع سيكون صفر)	4	__le32	خصائص ممتدة
UUID + رقم inode + رقم توليد inode + كتلة الدليل حتى المدخلة المزيفة التي تنطوي على حقل تدقيق المجموع	4	__le32	مدخلات أدلة
UUID + رقم inode + رقم توليد inode + جميع المدبات الصالحة + ذيل HTREE. (حقل تدقيق المجموع سيكون صفر)	4	__le32	عقد شجرة HTREE
UUID + رقم inode + رقم توليد inode + كامل كتلة المدى حتى حقل تدقيق المجموع.	4	__le32	مدبات
UUID + كامل المصفوفة الثنائية [32]. تدقيق المجاميع يخزن في group descriptor، ومقتطع في حالة كان حجم توصيف المجموعة 32 بايت (أي، 64bit^)	4 / 2	__le16 أو le32_	مصفوفات ثنائية
المعرف UUID + رقم inode + رقم توليد inode + كامل inode. (حقل تدقيق المجموع سيكون صفر) كل inode يملك تدقيق مجموع خاص.	4	__le32	مؤشرات الفهرسة
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.	2	__le16	توصيف المجموعات

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 جداً، لخفض التجزئة وفوقانية [15] البيانات الوصفية. 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 حجابات (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 extent[120]، مع أنه ليس واضح أين ذهبت تلك الرقع-التي تحولت أخيرا إلى " extent tree v2" تلك الشفرة لم تظهر منذ مايو 2015

البيانات المضمنة / البيانات المباشرة Inline Data

هذه الميزة تمكن بيانات الملف الصغيرة جدا من التناسب بسهولة داخل inode، (نظريا) هذا يخفض من استهلاك كتل القرص 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 وينتقل المحتوى إليها.

إن لم تستخدم الخصائص الممتدة سيكون بالمقدور 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.

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.

الملف المعزول / اليتيم (مقترح) Orphan File

هذا كان مقترح من جان كارا Jan Kara بداية عام 2015، هذه الميزة تهدف إلى خفض إقفال تنازع الموارد[37]locking contention أثناء عمليات الحذف باستبدال القائمة المتصلة البسطة للمؤشرات الفهرسة المعزولة (والقفل lock) بملف يتضمن عدة كتل. كل CPU يجب أن يكون قادر على المطالبة بكتلته الخاصة، مما يعني أنه يمكن تحديث لائحة المعزولات orphan list بدون اقفال.كل كتلة تتضمن قائمة من مؤشرات الفهرسة المعزولة[132] orphaned inodes؛ الاسترداد ينطوي على تكرار جميع كتل الملف المعزول orphan file بالبحث عن أرقام inodes غير الصفر لحذفها. هذه الميزة تأتي مع علم ميزة rocompat flag للإشارة إلى إمكانية استخدام الملف المعزول وعلم compat flag الذي يشير إلى أن الملف المعزول في الواقع يتضمن تسجيلات مؤشرات فهرسة معزولة orphaned inode records. صيغة كتلة الملف المعزول orphan file ستكون كالتالي:

حقل	حجم / نوع		وصف	
inodes	__le32[]	4	لائحة من مؤشرات الفهرسة المعزولة. القيمة 0 تعني أن المدخلة شاغرة.طول المصفوفة هو حجم الكتلة ناقص الذيل	
magic	__le32	4	الرقم السحري لكتل الملف المعزول،	0x0B10CA04
checksum	__le32	4	تدقيق مجموع كل من UUID + رقم inode + رقم توليد inode + الكتلة اليتيمة حتى التذيل (لكن بدون حاسبه)	

رقم مؤشر فهرسة الملف المعزول نفسه لم يتم تسويته بعد؛ اعتبارًا من مايو 2015 الرقع التجريبية تعيد استخدام مؤشر الفهرسة #9

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 inode الملف الاعتيادي، هذه"EA inode" ترتبط فقط[40] من جهة فهرس أسماء الخصائص الممتدة extended attribute name index ولا 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 inodes (أي عندما الملفات تتشارك 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 للبنية الواحدة inode (أي في الحالات التي لا يتم الرجوع فيها إلى EA inode من عدة inodes) للتأكد من أن مؤشر فهرسة EA inode هو الصحيح الذي يتم النفاذ إليه

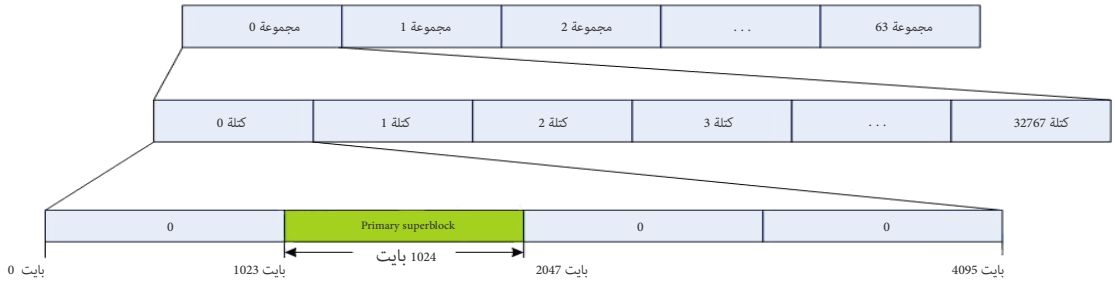
تقليديا توزيعات لينكس لا تملك شفرة إقلاع في نظام الملفات (وحدة التخزين) إلى جانب النواة. وتستخدم عوض ذلك شفرة الاقلاع الابتدائية في قطاع MBR. (راجع كتيب MBR و GPT)،

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!.

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

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 MFT في NTFS)، وتتضمن معلومات ضرورية لإقلاع نظام لينكس. لذلك توجد منها عدة نسخ احتياطية.لكن النسخة الأول فقط في أول كتلة من Block Group 0 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 [86][52]، 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 #1 block group، والأخرى في المجموعة الأخيرة في نظام الملفات. هذه الميزة الأخيرة تسمح بزيادة نسبة كتل البيانات المتماسة على القرص! للملفات. (راجع أيضا ميزة flex_bg)



مثال : 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

كتلة superblock تقع دائما عند بايت 1024 من بداية وحدة التخزين ودائما تكون 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 ستكون struct ext4_super_block (من بايت 1024 إلى بايت 2047 في بداية مجموعة الكتل 0) (مثال في نظام 32 بت)

```
dd if=/dev/sda1 bs=1024 count=1 skip=1 | hexdump -Cv
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEFF
0000 00 00 38 00 00 96 df 00 e6 2d 0b 00 22 68 48 00 |..8......-.."hH.|
0010 41 3a 33 00 00 00 00 00 02 00 00 00 02 00 00 00 |A:3.............|
0020 00 80 00 00 00 80 00 00 00 20 00 00 2a 73 ba 5a |......... ..*s.Z|
0030 09 7b be 5a 02 00 04 00 53 ef 01 00 01 00 00 00 |.{.Z....S.......|
0040 79 5e ba 5a 00 00 00 00 00 00 00 00 01 00 00 00 |y^.Z............|
0050 00 00 00 00 0b 00 00 00 00 01 00 00 3c 00 00 00 |............<...|
0060 42 02 00 00 7b 00 00 00 91 2b 02 40 5b 79 47 c2 |B...{....+.@[yG.|
0070 a4 8b 75 a6 e6 d5 cf f3 00 00 00 00 00 00 00 00 |..u.............|
0080 00 00 00 00 00 00 00 00 2f 00 61 72 67 65 74 00 |......../.arget.|
0490 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
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 fc 03 |................|
00d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00e0 08 00 00 00 00 00 00 00 00 00 00 00 96 63 e8 05 |.............c..|
00f0 1f d6 4f 28 8c 0d 03 33 f7 62 63 97 01 01 00 00 |..O(...3.bc.....|
0100 0c 00 00 00 00 00 00 00 d0 9d 37 58 0a f3 02 00 |..........7X....|
0110 04 00 00 00 00 00 00 00 00 00 00 00 ff 7f 00 00 |................|
0120 00 80 68 00 ff 7f 00 00 01 00 00 00 ff ff 68 00 |..h...........h.|
0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 |................|
0150 00 00 00 00 00 00 00 00 00 00 00 00 1c 00 1c 00 |................|
0160 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0170 00 00 00 00 04 00 00 00 86 4e d9 0a 00 00 00 00 |.........N......|
0180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
01a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
01b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
01c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
01d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
01e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
01f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0270 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 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
```

رمز تذكري	إزاحة	نوع	حجم				
s_inodes_count	0x00 (00)	__le32	4	عدد inodes الإجمالي في نظام الملفات [01] [56]			
s_blocks_count_lo	0x04 (04)	__le32	4	عدد الكتل الإجمالي في نظام الملفات [57]			
s_r_blocks_count_lo	0x08 (08)	__le32	4	عدد الكتل المحجوزة لمنع شغل نظام الملفات [58] (32 بت المنخفضة) (سيكون من تخصيص المستخدم الجذر؛ ذو الصلاحيات العليا) (أنظر للحبد 0x50)			
s_free_blocks_count_lo	0x0C (12)	__le32	4	عدد الكتل الحرة [59](الغير مخصصة)			
s_free_inodes_count	0x10 (16)	__le32	4	عدد inodes الحرة (الغير مخصصة) [60]			
s_first_data_block	0x14 (20)	__le32	4	أول كتلة بيانات (تتضمن Super Block) (ستكون 0 بالنسبة لجميع أحجام الكتلة، باستثناء كتلة 1 كيلوبايت التي يجب أن تكون 1 على الأقل) [61]			
s_log_block_size	0x18 (24)	__le32	4	حجم الكتلة [62]	2 ^ (10 + s_log_block_size)		
s_log_cluster_size	0x1C (28)	__le32	4	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			
s_blocks_per_group	0x20 (32)	__le32	4	عدد الكتل لكل مجموعة [63]			
s_clusters_per_group	0x24 (36)	__le32	4	عدد العناقيد لكل مجموعة، في حالة تمكين ميزة bigalloc. ما عدا ذلك. s_clusters_per_group يجب أن يساوي s_blocks_per_group			
s_inodes_per_group	0x28 (40)	__le32	4	عدد inodes لكل مجموعة [64]			
s_mtime	0x2C (44)	__le32	4	أختام زمنية	زمن وصل نظام الملفات (وحدة التخزين) آخر مرة بعدد الثواني (توقيت يونكس)		
s_wtime	0x30 (48)	__le32	4		زمن الكتابة إلى نظام الملفات آخر مرة بعدد الثواني (توقيت يونكس)		
s_mnt_count	0x34 (52)	__le16	2		عدد مرات وصل نظام الملفات (وحدة التخزين) منذ أخر فحص fsck (منذ استخدام أداة فحص تماسك / ثبات نظام الملفات!)		
s_max_mnt_count	0x36 (54)	__le16	2		عدد مرات وصل نظام الملفات المطلوبة قبل فحص fsck. (فحص تماسك نظام الملفات)		
s_magic	0x38 (56)	__le16	2	توقيع سحري! [03] (لتأكيد وجود نظام الملفات EXT2/3/4 على وحدة التخزين)	EXT2_SUPER_MAGIC	0xEF53	
s_state				(أعلام) حالة نظام الملفات [65] القيم الصالحة ستكون:			
				نظام الملفات نظيف (مفصول على نحو نظيف)	Unmounted cleanly	EXT4_VALID_FS	0x0001
				أخطاء في نظام الملفات	Errors detected	EXT4_ERROR_FS	0x0002
				استعادة inodes يتيمة(المعزولة)	Orphans being recovered	EXT4_ORPHAN_FS	0x0004
s_errors	0x3C (60)	__le16	2	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
s_minor_rev_level	0x3E (62)	__le16	2	مستوى مراجعة ثانوي (قيمة 16 بت) داخل مستوى المراجعة (راجع أيضا: نظام التحكم بالمراجعات)			
s_lastcheck	0x40 (64)	__le32	4	أختام زمنية	زمن الفحص الأخير، بعدد الثواني (توقيت يونكس)		
s_checkinterval	0x44 (68)	__le32	4		الفترة الأقصى بين الفحوص، بعدد الثواني		
s_creator_os				نظام التشغيل / هوية نظام التشغيل الذي عن طريقه تم إنشاء نظام الملفات على وحدة التخزين .			
				لينكس	Linux	EXT4_OS_LINUX	0
				حنو 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 الإصدار BSD4.4-Lite	Lites	EXT4_OS_LITES	4
s_rev_level	0x4C (76)	__le32	4	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
s_def_resuid	0x50 (80)	__le16	2	معرّف المستخدم، الذي يستطيع استخدام الكتل المحجوزة	نوع المعرّف : UID يشير إلى: المستخدم الجذري	EXT4_DEF_RESUID	0
s_def_resgid	0x52 (82)	__le16	2	معرّف المجموعة، التي تستطيع استخدام الكتل المحجوزة	نوع المعرّف : GID يشير إلى: المجموعة الجذرية	EXT4_DEF_RESGID	0
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							
s_first_ino	0x54 (84)	__le32	4	أول inode غير محجوز في نظام الملفات[67]			
s_inode_size	0x58 (88)	__le16	2	حجم بنية inode- بعدد بايتات [04] [68] (راجع فقرة: Inode Size)			
s_block_group_nr	0x5A (90)	__le16	2	رقم مجموعة الكتل التي تنتمي إليها هذه الكتلة العليا superblock -[69]			
s_feature_compat	0x5C (92)	__le32	4	مجموعة أعلام الميزّات المتوافقة [70] [85]	Compatible feature set flags :		
				كتل الدليل المخصصة مسبقا من أجل خفض التجزئة (عند إنشاء دليل جديد) (أنظر للحبد 0xCD)	COMPAT_DIR_PREALLOC	0x0001	

				0x0002	COMPAT_IMAGIC_INODES	"imagic inodes" ليس واضح ماذا يفعل هذا العلم (لكنه يشير إلى وجود inodes خادوم AFS)
				0x0004	COMPAT_HAS_JOURNAL	نظام ملفات مزود بقيد حوادث (أي الإجراء يرتكز على سجل دوري) [113]
				0x0008	COMPAT_EXT_ATTR	دعم الخصائص الممتدة (inodes تملك خصائص ممتدة)
				0x0010	COMPAT_RESIZE_INODE	يملك كتل GDT من أجل توسع نظام الملفات (نظام الملفات يستطيع إعادة تحجيم نفسه إلى أقسام أكبر) [50] [95]
				0x0020	COMPAT_DIR_INDEX	يملك فهارس للدليل [40] (تستخدم شجرة hashed b-trees لتسريع عمليات البحث في الأدلة الكبيرة)
				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" (غير مستخدمة)
				0x0100	COMPAT_EXCLUDE_BITMAP	للإشارة إلى وجود exclude bitmaps ذات الصلة بالصور snapshot -[29] (غير مستخدمة 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]
s_feature_incompat	0x60 (96)	__le32	4	مجموعة أعلام الميزّات الغير متوافقة [71] [85] : Incompatible feature set		
				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 تستخدم مديات في نظام الملفات (دعم المديات extents)
				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	بيانات في inode
				0x10000	INCOMPAT_ENCRYPT	وجود inodes مشفرة على نظام الملفات
s_feature_ro_compat	0x64 (100)	__le32	4	مجموعة أعلام الميزّات المتوافقة – في وضعية القراءة فقط [27] [72][85] : Readonly-compatible feature set		
				0x0001	RO_COMPAT_SPARSE_SUPER	توصيف المجموعات ونسخ الكتلة العليا ستكون متناثرة Sparse superblocks (أي ليست في كل المجموعات) [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 مستخدمة في النواة أو حزمة e2fsprogs)
				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	توصيف المجموعات Group descriptors يملك تدقيق مجامع [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]
				0x0100	RO_COMPAT_QUOTA	تمكين نظام الحصة (الحصص النسبية للقرص) QUOTA [13]
				0x0200	RO_COMPAT_BIGALLOC	نظام الملفات يدعم bigalloc، هذا يعني تعقب مديات 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_PROJECT	نظام الملفات يتعقب حصص القرص باستخدام project quotas (هذا نوع جديد من الحصص !) [14]
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 ضغط البيانات. قيمة 32 بت من أجل تحديد طريقة للضغط 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 قيْد الحوادث journal device في حالة تعيين علم ميزة قيد الحوادث الخارجي (قيمة 32 بت)
s_last_orphan	0xE8 (232)	__le32	4	بداية لائحة inodes اليتيمة (أو المعزولة) من أجل الحذف [73]	
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	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 توصيف 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	تنفيذ جميع البيانات والبيانات الوصفية إلى قيد الحوادث All data and metadata are committed to the journal
				0x0040	EXT4_DEFM_JMODE_ORDERED	تخليص جميع البيانات (من الصوان) إلى القرص قبل تنفيذ البيانات الوصفية إلى قيد الحوادث.
				0x0060	EXT4_DEFM_JMODE_WBACK	ترتيب البيانات غير محفوظ(غير محمي)؛ يمكن كتابة البيانات بعد كتابة البيانات الوصفية.
				0x0100	EXT4_DEFM_NOBARRIER	تعطيل كتابات الصوان إلى القرص write flushes (راجع آلية حواجز الكتابة BARRIER [134] 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]
				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 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 للنظام snapshot
				0x0020	EXT2_FLAGS_FIX_SNAPSHOT	Snapshot inodes corrupted	inodes صور Snapshot فاسدة
				0x0040	EXT2_FLAGS_FIX_EXCLUDE	Exclude bitmaps corrupted	مصفوفات ثنائية فاسدة للاستثناء الصور Exclude bitmaps
s_raid_stride	0x164 (356)	__le16	2	وحدة شريطية في مصفوفة رايد 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 رايد 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 الكتل على أقراص SSD (نتيجة دورات المسح المحدودة P/E cycles)			
s_snapshot_inum	0x180 (384)	__le32	4	صور زمنية انتقائية للنظام snapshot		رقم inode الصورة النشطة. [29] (غير مستخدم في e2fsprogs / لينكس) Inode number of active snapshot	
s_snapshot_id	0x184 (388)	__le32	4			هوية تتابعية للصورة النشطة (غير مستخدم في e2fsprogs / لينكس) sequential ID of active snapshot	
s_snapshot_r_blocks_count	0x188 (392)	__le64	8			عدد الكتل المحجوزة للصورة النشطة للاستعمال مستقبلا (غير مستخدم في e2fsprogs / لينكس)	
s_snapshot_list	0x190 (400)	__le32	4			رقم inode بداية لائحة الصور snapshot على القرص. (غير مستخدم في e2fsprogs / لينكس)	
s_error_count	0x194 (404)	__le32	4	الأخطاء مثال: [127]	number of fs errors seen	عدد الأخطاء المنظورة	
s_first_error_time	0x198 (408)	__le32	4		first time an error happened	زمن وقوع أول خطأ، بعدد الثواني (توقيت يونكس)	
s_first_error_ino	0x19C (412)	__le32	4		inode involved in first error	inode المرتبط بأول خطأ	
s_first_error_block	0x1A0 (416)	__le64	8		block involved of first error	رقم الكتلة المرتبطة بأول خطأ	
s_first_error_func[32]	0x1A8 (424)	__u8	32		function where the error happened	اسم الوظيفة أين وقع الخطأ	
s_first_error_line	0x1C8 (456)	__le32	4		line number where error happened	رقم السطر أين وقع الخطأ	
s_last_error_time	0x1CC (460)	__le32	4		most recent time of an error	زمن أحدث خطأ، بعدد الثواني (توقيت يونكس)	
s_last_error_ino	0x1D0 (464)	__le32	4		inode involved in last error	inode المرتبط بأحدث خطأ	
s_last_error_line	0x1D4 (468)	__le32	4		line number where error happened	رقم السطر أين وقع أحدث خطأ	
s_last_error_block	0x1D8 (472)	__le64	8		block involved of last error	رقم الكتلة المرتبطة بأحدث خطأ	
s_last_error_func[32]	0x1E0 (480)	__u8	32		function where the error happened	اسم الوظيفة أين وقع أحدث خطأ	
s_mount_opts[64]	0x200 (512)	__u8	64	سلسلة ASCIIZ (ترميز أحرف) من أجل خيارات وصل نظام الملفات [34]			
s_usr_quota_inum	0x240 (576)	__le32	4	حصص القرص [13]	inode number of user quota file	رقم inode ملف حصة المستخدم	
s_grp_quota_inum	0x244 (580)	__le32	4		inode number of group quota file	رقم inode ملف حصة المجموعة	
s_overhead_blocks	0x248 (584)	__le32	4	العناقيدأو الكتل الفوقانية [15] في نظام الملفات fs overhead blocks/clusters in (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 هذا) التي قد يصل عددها عند الاستخدام إلى 4 خوارزميات في أي وقت؛ شفرات الخوارزميات الصالحة مع أنماطها ستكون كالتالي:			
				0	EXT4_ENCRYPTION_MODE_INVALID		خوارزمية غير صالحة
				1	EXT4_ENCRYPTION_MODE_AES_256_XTS		AES إيه إي إس 256-بت في نمط XTS
				2	EXT4_ENCRYPTION_MODE_AES_256_GCM		AES إيه إي إس 256-بت في نمط GCM
				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 من أجل خوارزمية string2key / string-to-key (للتشفير) Salt used for string2key algorithm			
s_lpf_ino	0x268 (616)	__le32	4	موقع / رقم inode دليل lost+found			
s_prj_quota_inum	0x26C (620)	__le32	4	inode الذي يتعقب حصص القرص من نوع project quotas. [14]			
s_checksum_seed	0x270 (624)	__le32	4	crc32c(~0, $orig_fs_uuid)		بذرة تدقيق المجموع [24] المستخدمة في حسابات metadata_csum.هذه القيمة في حالة تعيين csum_seed	
s_reserved[98]	0x274 (628)	__le32	392	حشو إلى نهاية الكتلة [98]			
s_checksum	0x3FC (1020)	__le32	4	crc32c(superblock)		تدقيق مجموع Super block (حساب تدقيق مجموع بنية الكتلة العليا superblock يشمل أيضا FS UUID)	

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

هذه المصفوفة تستخدم في تحديد معاملات جميع مجموعات الكتل، كل مجموعة على نظام الملفات يرتبط بها واصف [16] وكما تظهر في التخطيط، هذه المصفوفة (إن وجدت) ستكون العنصر الثاني داخل المجموعة.

كل مجموعة في إعداداتها المعيارية، 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 والتي ستكون الكتلة الثالثة في نظام ملفات كتلة 1 كيلوبايت، أو الكتلة الثانية في نظام ملفات كتلة 2 كيلوبايت والأكبر.

لاحظ كيف يسجل توصيف المجموعة 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 المجموعة. آلية flex_bg تستغل هذه المميزة في جمع عدة مجموعات في مجموعة مرنة واحدة ووضع inode tables و inode bitmaps و block bitmaps من كل المجموعات في أول مجموعة من flex group.

وفي حالة تعيين ميزة meta_bg، [95] يتم جمع عدة مجموعات كتل في مجموعة وصفية واحدة meta group. لكن فقط مجموعة الكتل الأولى والثاني والأخيرة في المجموعة الوصفية الأكبر تتضمن واصفات المجموعة من أجل المجموعات داخل meta group.. لحظ أن المميزتان flex_bg و meta_bg لا تبدوان متنافيتان.(أي يمكن أن تقع في الوقت نفسه)

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، توصيف مجموعة الكتل يتمدد إلى 64 بايت على الأقل؛ هذا الحجم يخزن في superblock. كل مجموعة كتل ستملك البنية التالية في توصيف مجموعة الكتل. بينما كل جدول توصيف مجموعات كتل، سيتضمن مدخلات (معلومات) عن جميع مجموعات الكتل.

توصيف أو واصف مجموعة الكتل block group descriptor في struct ext4_group_desc (مدخلة لكل مجموعة كتل)

```
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.|
1000
```

رمز تذكري	إزاحة	نوع	حجم			
bg_block_bitmap_lo	0x00 (00)	__le32	4	عنوان الكتلة / موقع المصفوفة الثنائية من أجل الكتل (32-بت المنخفضة) Blocks bitmap block		
bg_inode_bitmap_lo	0x04 (04)	__le32	4	عنوان الكتلة / موقع المصفوفة الثنائية من أجل inodes [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	تَعْداد الكتل الحرة (16-بت المنخفضة)		
bg_free_inodes_count_lo	0x0E (14)	__le16	2	تَعْداد inodes الحرة (16-بت المنخفضة)		
bg_used_dirs_count_lo	0x10 (16)	__le16	2	تَعْداد الأدلة في مجموعة الكتل (16-بت المنخفضة) أي عدد inodes المخصص للأدلة Directories count		
bg_flags	0x12 (18)	__le16	2	أعلام مجموعة الكتل Block group flags :		
				0x0001	EXT4_BG_INODE_UNINIT	Inode table/bitmap not initialized — المصفوفة الثنائية وجدول inodes غير مهيئة
				0x0002	EXT4_BG_BLOCK_UNINIT	Block bitmap not initialized — المصفوفة الثنائية للكتل غير مهيئة
				0x0004	EXT4_BG_INODE_ZEROED	On-disk itable initialized to zero — جدول inodes مصفر (ITABLE_ZEROED)
bg_exclude_bitmap_lo	0x14 (20)	__le32	4	موقع المصفوفة الثنائية "Exclude bitmap" لإقصاء! الصور snapshot ـ (32-بت المنخفضة) Exclude bitmap for snapshots		
bg_block_bitmap_csum_lo	0x18 (24)	__le16	2	crc32c(s_uuid+grp_num+bbitmap)		تدقيق مجموع المصفوفة الثنائية للكتل (16-بت المنخفضة)
bg_inode_bitmap_csum_lo	0x1A (26)	__le16	2	crc32c(s_uuid+grp_num+ibitmap)		تدقيق مجموع المصفوفة الثنائية inodes (16-بت المنخفضة)
bg_itable_unused_lo	0x1C (28)	__le16	2	تَعْداد inodes الغير مستخدمة (16-بت المنخفضة) [91]		
bg_checksum	0x1E (30)	__le16	2	crc16(sb_uuid+group+desc)		تدقيق مجموع توصيف المجموعة (أنظر [92]) Group descriptor checksum
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 inodes ـ (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	تَعْداد الكتل الحرة (16-بت العليا)		
bg_free_inodes_count_hi	0x2E (46)	__le16	2	تَعْداد inodes الحرة (16-بت العليا)		
bg_used_dirs_count_hi	0x30 (48)	__le16	2	تَعْداد الأدلة (16-بت العليا)		
bg_itable_unused_hi	0x32 (50)	__le16	2	تَعْداد inodes الغير مستخدمة (16-بت العليا)		
bg_exclude_bitmap_hi	0x34 (52)	__le32	4	موقع كتلة المصفوفة الثنائية "Exclude bitmap" لإقصاء! الصور snapshot ـ (32-بت العليا) Exclude bitmap block		
bg_block_bitmap_csum_hi	0x38 (56)	__le16	2	crc32c(s_uuid+grp_num+bbitmap)		تدقيق مجموع المصفوفة الثنائية للكتل (16-بت العليا)
bg_inode_bitmap_csum_hi	0x3A (58)	__le16	2	crc32c(s_uuid+grp_num+ibitmap)		تدقيق مجموع المصفوفة الثنائية inodes ـ (16-بت العليا)
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، (أي تدقيق مجموع توصيف المجموعات) وتعطيل ميزة metadata_csum (أي تدقيق مجموع البيانات الوصفية)، تدقيق مجموع مجموعة الكتل سيكون crc16 بحساب: معرف FS UUID. ورقم المجموعة، وبنية توصيف المجموعة. وفي حالة تعيين ميزة metadata_csum، تدقيق مجموع مجموعة الكتل سيكون 16 بت المنخفضة بحساب تدقيق مجموع: معرف FS UUID. ورقم المجموعة، وبنية توصيف المجموعة. تدقيق مجموع كل من المصفوفات الثنائية inode bitmap و block bitmap سيكون بحساب: معرف 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.

بمعني 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 [103] e2fsprogs 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 خَرْج 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 8 * number_of_bytes_in_a_logical_block بالنسبة للكتل كل بت يمثل الوضعية 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 618 (+618), Inode bitmap at 634 (+634) إذن
Block bitmap : 618 × 4096 = 2531328
Inode bitmap : 634 × 4096 = 2596864
# dd if=/dev/sda1 bs=4096 skip=618 count=1 | hexdump -Cv  أو
# od -w32 -N 4096 -j 2531328 -A d -t x1 -v group0.dd (file)
```

راجع صفحة دليل (يونكس) "man od"

```
$ 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
2535424
```

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

وفي تمثيل 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 الفعلي لكل مجموعة (جرب: "Inodes per group" -i | grep /dev/xxxx -l 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 :

```
# blkcat -f linux-ext3 ext3.dd 2 | xxd
0000: ffff ffff ffff ffff ffff ffff ffff ffff ................
[Repeated]
1168: ff01 fcff ffff 0ffe ffff ffff 03fe ff f ................
```

← مستوى بايت

9,359	9,358	9,357	9,356	9,355	9,354	9,353	9,352

← جزء من مصفوفة الكتل

0	0	0	0	0	0	0	1

← مستوى بت

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

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][115][116]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، والإزاحة 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 00 02 00 00 00 48 12 00 00 04 00 00 |.........H......|
0050 23 03 00 00 00 e0 12 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 |................|
0100
```

حجم	نوع	إزاحة	رمز تذكري				
2	__le16	0x00 (00)	i_mode	نمط الملف (قيمة 16 بت) هذه ستكون خصائص معيارية للملف في شكل أذون / أنماط / أنواع / أعلام / قيم			
				الآخرون يمكنهم التنفيذ / البحث	S_IXOTH	0x1	أذون نفاذ / حقوق النفاذ
				الآخرون يمكنهم الكتابة	S_IWOTH	0x2	
				الآخرون يمكنهم القراءة	S_IROTH	0x4	
				أعضاء المجموعة يمكنهم التنفيذ / البحث	S_IXGRP	0x8	
				أعضاء المجموعة يمكنهم الكتابة	S_IWGRP	0x10	
				أعضاء المجموعة يمكنهم القراءة	S_IRGRP	0x20	
				المالك يمكنه التنفيذ / البحث	S_IXUSR	0x40	
				المالك يمكنه الكتابة	S_IWUSR	0x80	
				المالك يمكنه القراءة	S_IRUSR	0x100	
				بت تقييد (Sticky bit) ! [17]	S_ISVTX	0x200	أعلام أنماط خاصة (تنفيذية)
				تعيين معرف المجموعة (GID)	S_ISGID	0x400	
				تعيين معرف المستخدم (UID)	S_ISUID	0x800	
				أنبوبة اتصال مسماة	S_IFIFO	0x1000	صيغة ملف/ أنواع ملفات متنافية [قيم يجب تعين واحد منها فقط] [33]
				جهاز محرفي	S_IFCHR	0x2000	
				دليل (مجلد)	S_IFDIR	0x4000	
				جهاز كتل (جهاز كتلي)	S_IFBLK	0x6000	
				ملف اعتيادي	S_IFREG	0x8000	
				وصلة رمزية (وصلة لينة) [111]	S_IFLNK	0xA000	
				مقبس	S_IFSOCK	0xC000	
2	__le16	0x02 (02)	i_uid	معرف المالك UID المرتبط بالملف -(16-بت المنخفضة)			
4	__le32	0x04 (04)	i_size_lo	حجم الملف بالبايت (32-بت المنخفضة) [75]			
4	__le32	0x08 (08)	i_atime	زمن النفاذ آخر مرة إلى نظام الملفات، بعدد الثواني (توقيت يونكس)	أو سيكون تدقيق مجموع قيمة الخاصية الممتدة في حالة EA_INODE		أختام زمنية أو قيمة الخاصية الممتدة [76]
4	__le32	0x0C (12)	i_ctime	زمن تغير آخر مرة مؤشر الفهرسة (نفسه) بعدد الثواني (توقيت يونكس)	أو 32 بت المنخفضة للتعداد المرجعي للقيمة الخاصة في حالة EA_INODE		
4	__le32	0x10 (16)	i_mtime	زمن تعديل البيانات آخر مرة، بعدد الثواني (توقيت يونكس)	أو رقم مؤشر الفهرسة الذي يمتلك الخاصية الممتدة في حالة EA_INODE		
4	__le32	0x14 (20)	i_dtime	زمن الحذف، بعدد الثواني (توقيت يونكس). قيمة 32 بت تشير إلى زمن حذف مؤشر الفهرسة			
2	__le16	0x18 (24)	i_gid	معرف المجموعة GID (16-بت المنخفضة)			
2	__le16	0x1A (26)	i_links_count	تَعْداد الروابط الصلبة (عدد الروابط إلى مؤشر الفهرسة / الملف) [77] [111]			

i_blocks_lo	0x1C (28)	__le32	4	تَعْداد الكتل"block"ـ(32-بت المنخفضة) (عدد الكتل المحجوزة للملف. قد تعني حجم الكتلة بالقطاع مثل 512 بايت أو بالكتلة مثل 4096 بايت بتعيين HUGE_FILE) [78]				
i_flags	0x20 (32)	__le32	4	أعلام مؤشرات الفهرسة (قيمة 32 بت تشير إلى كيف سيتصرف تطبيق ext4 عند النفاذ إلى بيانات مؤشر الفهرسة هذا) : Inode flags				
				هذا الملف يستلزم الحذف الأمن "secure deletion" (غير مطبق)	EXT4_SECRM_FL	0x00000001		
				يجب حفظ هذا الملف (record for undelete)، (غير مطبق)	EXT4_UNRM_FL	0x00000002		
				الملف مضغوط (ليس مطبق بالضرورة)	EXT4_COMPR_FL	0x00000004		
				كافة الكتابات إلى الملف يجب أن تكون متزامنة (synchronous updates)	EXT4_SYNC_FL	0x00000008		
				الملف ثابت، غير قابل للتغير	EXT4_IMMUTABLE_FL	0x00000010		
				يمكن فقط إلحاق الكتابات بالملف (append only)	EXT4_APPEND_FL	0x00000020		
				وسيلة (1) dump لا يجب أن تطرح هذا الملف (لا يجب حفظ الملفات عند استخدام dump في نسخ الملفات الاحتياطية)	EXT4_NODUMP_FL	0x00000040		
				لا تجدد زمن النفاذ (i_atime)	EXT4_NOATIME_FL	0x00000080		
				ملف مضغوط معدل [26] (غير مستخدم) (Dirty compressed file)	EXT4_DIRTY_FL	0x00000100		
				الملف يملك عنقود مضغوط أو أكثر (غير مستخدم) (compressed blocks)	EXT4_COMPRBLK_FL	0x00000200		
				لا تضغط الملف (النفاذ إلى بيانات مضغوطة خام/أولية) (غير مستخدم)	EXT4_NOCOMPR_FL	0x00000400		
				مؤشر فهرسة مشفر. قيمة بت هذه كانت سابقا EXT4_ECOMPR_FL لأجل (خطأ ضغط) ولم يستخدم أبدا	EXT4_ENCRYPT_FL	0x00000800		
				الدليل يملك فهارس هاش [31] hashed indexes (أي الدليل يستخدم شجرة HTree)	EXT4_INDEX_FL	0x00001000		
				الدليل السحري magic directory في AFS	EXT4_IMAGIC_FL	0x00002000		
				يجب دائما كتابة بيانات الملف من خلال قيد الحوادث [113]	EXT4_JOURNAL_DATA_FL	0x00004000		
				لا يجب دمج ذيل الملف File tail (الجزء الأخير من الملف) (غير مستخدم في ext4)	EXT4_NOTAIL_FL	0x00008000		
				يجب كتابة جميع بيانات مدخلة الدليل بالتزامن (أنظر سلوك dirsync) (الأدلة فقط)	EXT4_DIRSYNC_FL	0x00010000		
				قمة هرميّة الأدلة Top of directory hierarchies	EXT4_TOPDIR_FL	0x00020000		
				هذا ملف ضخم huge file (يعين إلى كل ملف ضخم)ـ[54]	EXT4_HUGE_FILE_FL	0x00040000		
				مؤشر الفهرسة يستخدم المديات extents	EXT4_EXTENTS_FL	0x00080000		
				مؤشر الفهرسة يخزن في كتل بياناته قيمة خاصة ممتدة كبيرة large EA	EXT4_EA_INODE_FL	0x00200000		
				هذا الملف يملك كتل مخصصة تتجاوز نهاية الملف EOF (مهجور)	EXT4_EOFBLOCKS_FL	0x00400000		
				لا تستخدم عملية Copy-on-write على الملف Do not cow file	FS_NOCOW_FL	0x00800000		
				مؤشر فهرسة عبارة عن صورة snapshotـ[29] (ليس في مستودع المشروع mainline)	EXT4_SNAPFILE_FL	0x01000000		
				حذف الصورة snapshot (ليس في mainline)	EXT4_SNAPFILE_DELETED_FL	0x04000000		
				اكتمل تقليص الصورة snapshotـ (ليس في mainline)	EXT4_SNAPFILE_SHRUNK_FL	0x08000000		
				مؤشر فهرسة يملك بيانات مضمنة inline data	EXT4_INLINE_DATA_FL	0x10000000		
				إنشاء children بنفس هوية المشروع (ملفات فرعية بنفس هوية الدليل الأم) Create with parents projid [14]	EXT4_PROJINHERIT_FL	0x20000000		
				محجوز من أجل مكتبة ext4 library	EXT4_RESERVED_FL	0x80000000		
				أعلام مرئية للمستخدم-	EXT4_FL_USER_VISIBLE	0x204BDFFF	أعلام مُجمَلة	
				أعلام تقبل التعديل من المستخدم [79]	EXT4_FL_USER_MODIFIABLE	0x204B80FF		
i_osd1	0x24 (36)	__le32	4	مخصص للنظام التشغيل (قيمة 32 بت) Union osd1 (OS dependent 1)				
				وصف	حجم / نوع	إزاحة	رمز تذكري	وسم
				إصدارة مؤشر الفهرسة.[80] (هذا الحقل كان سابقا l_i_reserved1)	4 __le32	0x00	l_i_version	linux1
				؟ "translator“	4 __le32	0x00	h_i_translator	hurd1
				محجوز	4 __le32	0x00	m_i_reserved	masix1
i_block[EXT4_N_BLOCKS=15]	0x28 (40)		60	شجرة مديات في ext4 (راجع "inode.i_block") أو ستكون مخطط ربط الكتل (أي مؤشرات Pointers إلى الكتل) (15 × “32-بت”) [97] [100]				
i_generation	0x64 (100)	__le32	4	إصدارة الملف أو Generation number (مستخدم من قبل بروتوكول نظام الملفات الشبكي NFS)ـ[106]				
i_file_acl_lo	0x68 (104)	__le32	4	كتلة الخاصة الممتدة (32-بت المنخفضة)، و ACLs ستكون إحدى هذه الخصائص الممتدة الكثيرة؛ اسم هذا الحقل ربما كان نتيجة أول استخدام للخصائص الممتدة في ACLs.				
i_size_high / i_dir_acl	0x6C (108)	__le32	4	حجم الدليل/ الملف (32-بت العليا). اسم هذا الحقل في EXT2/3 هو i_dir_acl، رغم ذلك يعين إلى الصفر ولا يستخدم أبدا [09] [81]				
i_obso_faddr	0x70 (112)	__le32	4	عنوان كتلة fragment (حقل ملغى في ext4)				
i_osd2	0x74 (116)		12	مخصص للنظام التشغيل (قيمة 96 بت) Union osd2 (OS dependent 2)				
				وصف	حجم / نوع	إزاحة	رمز تذكري	وسم
				تَعْداد الكتل (16-بت العليا) (راجع الملاحظة الملحقة بـ i_blocks_lo)ـ[21]	2 __le16	0x00 (00)	l_i_blocks_high	linux2
				كتلة الخاصة الممتدة (16-بت العليا)، تاريخيا موقع أذون الملف ACL، (راجع الخصائص الممتدة)	2 __le16	0x02 (02)	l_i_file_acl_high	

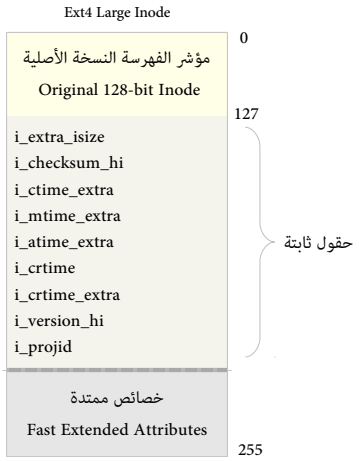
معرف المستخدم المالك UID ـ(16-بت العليا) [45]		2	__le16	0x04 (04)	l_i_uid_high				
معرف المجموعة GID ـ(16-بت العليا) [45]		2	__le16	0x06 (06)	l_i_gid_high				
تدقيق مجموع مؤشر الفهرسة (16-بت العليا)	crc32c(uuid+inum+inode)	2	__le16	0x08 (08)	l_i_checksum_lo				
غير مستعمل		2	__le16	0x0A (10)	l_i_reserved				
محجوز ؟ [21]		2	__le16	0x00 (00)	h_i_reserved1	Hurd2			
نمط الملف ـ(16-بت العليا)		2	__le16	0x02 (02)	h_i_mode_high				
معرف المستخدم المالك UID ـ(16-بت العليا) [45]		2	__le16	0x04 (04)	h_i_uid_high				
معرف المجموعة GID ـ(16-بت العليا) [45]		2	__le16	0x06 (06)	h_i_gid_high				
هوية مؤلّف الملف ! أو Author code ؟ـ [82]		4	__u32	0x08 (08)	h_i_author				
محجوز [21]		2	__le16	0x00 (00)	h_i_reserved1	masix2			
كتلة الخاصة 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)		2	__le16	0x80 (128)	i_extra_isize				
تدقيق مجموع مؤشر الفهرسة (16-بت العليا)	crc32c(uuid+inum+inode)	2	__le16	0x82 (130)	i_checksum_hi				
بتات زمن التغير (حقول Extra) هذا يوفر (فصل زمني) دقة أقل من الثانية sub-second (راجع الأختام الزمنية في inode)	أختام زمنية تستخدم نانو ثانية	4	__le32	0x84 (132)	i_ctime_extra				
بتات زمن التعديل (حقول Extra) هذا يوفر دقة أقل من الثانية		4	__le32	0x88 (136)	i_mtime_extra				
بتات زمن النفاذ (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 ملفات ext2/3، حجم بنية inode[116][115] كان ثابت عند 128 بايت (EXT2_GOOD_OLD_INODE_SIZE) مع حجم تسجيلة قرص أيضا 128 بايت. لكن مع بدأ استخدام ext4 صار بالإمكان في زمن التهيئة تخصيص inodes بحجم أكبر على القرص، 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 الفعلي المستخدم من struct ext4_inode خلف (128-بايت الأصلية) ext2 inode يسجل في حقل i_extra_isize، هذا يسمح بنمو struct ext4_inode مع النواة الجديدة دون الحاجة إلى ترقية جميع inodes على القرص.النفاذ إلى الحقول التي خلف EXT2_GOOD_OLD_INODE_SIZE ينبغي أن يتحقق ضمن i_extra_isize.

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



تحديد موقع مؤشر الفهرسة

Finding an Inode

مؤشرات الفهرسة [115][116] ستكون بترتيب عددي. كل inode number (مؤشر) في جدول inode table يشير إلى بنية Inode. حجم الجدول يحدد زمن التهيئة لاحتواء أقصى عدد من المدخلات.

كل مجموعة كتل تتضمن عدد من Inodes. يحدد في 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. بعد تحديد هوية الكتلة، يمكن إيجاد Inode (المحلي) المطلوب ضمن inode table في مجموعة الكتل باستخدام الصيغة: index = (inode_num - 1) % sb->s_inodes_per_group. وللحصول على عنوان بايت (الإزاحة) ضمن inode table، نستخدم: offset = index * sb->s_inode_size..

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

عينية من حساب مؤشر فهرسة

s_inodes_per_group = 1712

شرح	قاعدة	نتيجة	مثال
حجم inode table (بايتات) (الذي يكفي لتخزين على الأقل القيمة):	sb.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	$ 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، أوست عشري 16 في ibase، وخرج عشري 10، أو ست عشري 16 في obase

الأختام الزمنية في مؤشر الفهرسة

Inode Timestamps

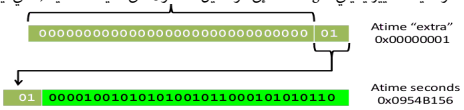
أربعة أختام زمنية (أو بصمات وقت) مسجلة في 128 بايت المنخفضة من بنية Inode هي زمن تغير مؤشر الفهرسة ctime، زمن النفاذ atime، زمن تعديل البيانات mtime، وزمن الحذف dtime. هذه الحقول الأربعة، أعداد صحيحة 32-بت تحمل إشارة [128] وتمثل الثواني منذ توقيت يونكس(1970-01-01 00:00:00 (GMT أو (@000000000 -d date)، هذا يعني أن الحقول ستطفو ($2^{31} - 1$) في يناير 2038. بالنسبة للمؤشرات الفهرسة التي ليس لها رابط من أي دليل لكنها ما تزال مفتوحة (orphan inodes)، حقل dtime سيفيض لاستخدامه مع لائحة المعزولة orphan list. حقل s_last_orphan في superblock سيشير إلى أول Inode في لائحة المعزولة؛ حينذاك dtime سيكون رقم مؤشر الفهرسة المعزول orphaned inode التالي، أو سيكون صفر إذا لم تكن هناك معزولات أخرى orphans.

حقول ctime وatime و mtime ستتمدد إلى 64 بت إذا كان حجم بنية مؤشر الفهرسة sb->s_inode_size أكبر من 128 بايت وحقل i_inode_extra يكفي للتطويق i_[cma]time_extra في حقل "extra"_(32-بت) تستخدم 2 بت المنخفضة لمد حقل الثواني من 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.

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.

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:

نطاق زمني صالح valid time range	tv_sec ـ64-بت المترجمة 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
1901-12-13 إلى 31-12-1969	-0x80000000 - 0x00000001	0	1	0 0
1970-01-01 إلى 19-01-2038	0x000000000 - 0x07fffffff	0	0	0 0
2038-01-19 إلى 07-02-2106	0x080000000 - 0x0ffffffff	0x100000000	1	0 1
2106-02-07 إلى 25-02-2174	0x100000000 - 0x17fffffff	0x100000000	0	0 1
16-03-2242 إلى 2174-02-25	0x180000000 - 0x1ffffffff	0x200000000	1	1 0
04-04-2310 إلى 2242-03-16	0x200000000 - 0x27fffffff	0x200000000	0	1 0
22-04-2378 إلى 2310-04-04	0x280000000 - 0x2ffffffff	0x300000000	1	1 1
10-05-2446 إلى 2378-04-22	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 من 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 2310.

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

العنونة المباشرة والغير مباشرة للكتل

Direct/Indirect Block Addressing

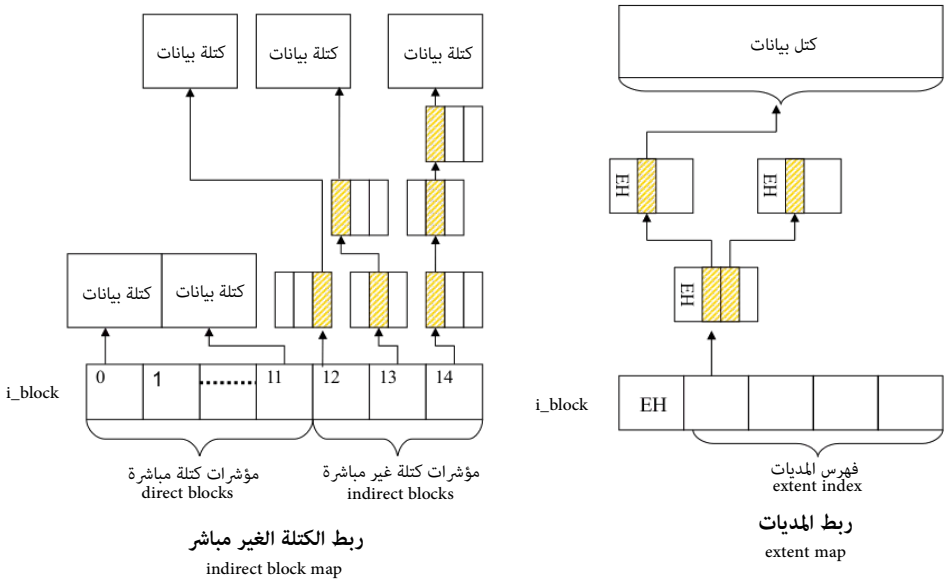
في أنظمة ملفات ext2/3، أرقام كتل الملف تعين إلى أرقام الكتل المنطقية عبر ربط للكتل 1-1 (قد يصل إلى) ثلاث مستويات. لإيجاد الكتلة المنطقية التي تخزن كتلة الملف، الشفرة ستبحث في كامل هذه البنية المعقدة على نحو متزايد [97] لاحظ هنا، لا يوجد رقم سحري أو تدقيق مجموع يضمن خلاء الكتلة من ما يسمى القمامة garbage.

<table>
<tr><td colspan="4">تشير إلى</td><td>إزاحة i.i_block</td></tr>
<tr><td colspan="4">تعين (أو ربط) مباشر إلى كتل الملف من 0 إلى 11 (تعيين أول 12 كتلة من داخل inode)</td><td>0 إلى 11</td></tr>
<tr><td colspan="4">كتلة غير مباشرة : (كتل الملف من 12 إلى (4 / $block_size$) + 11 أو من 12 إلى 1035 في حالة كتل 4 كيلوبايت أي 4096 = block_size)</td><td rowspan="3">12</td></tr>
<tr><td colspan="2">تشير إلى</td><td>إزاحة كتلة غير مباشرة</td></tr>
<tr><td colspan="2">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 colspan="4">كتلة غير مباشرة مزدوجة: (كتل الملف من 12 + $block_size/4 إلى 11 + (4 / $block_size$) + 2 ^ (4 / $block_size$) أي من 1036 إلى 1049611 في حالة كتل 4 كيلوبايت)</td><td rowspan="4">13</td></tr>
<tr><td colspan="2">تشير إلى</td><td>إزاحة كتلة غير مباشرة مزدوجة</td></tr>
<tr><td colspan="2">تعيين 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>تعيين مباشرة إلى كتل (4 / $block_size$) (1024 في حالة كتل 4 كيلوبايت)</td><td>من 0 إلى (4 / $block_size$)</td></tr>
<tr><td colspan="4">كتلة غير مباشرة ثلاثية: (كتل الملف من 12 + (4 / $block_size$) + 2 ^ (4 / $block_size $) إلى 12 + (4 / $block_size$) + 2 ^ (4 / $block_size$) + 3 ^ (4 / $block_size$) أي من 1049612 إلى 1074791436 في حالة كتل 4k)</td><td rowspan="6">14</td></tr>
<tr><td colspan="2">تشير إلى</td><td>إزاحة كتلة غير مباشرة ثلاثية</td></tr>
<tr><td colspan="2">تعين إلى الكتل الغير مباشرة المزدوجة (4 / $block_size$) (1024 في حالة كتل 4 كيلوبايت)</td><td rowspan="4">من 0 إلى (4 / $block_size$)</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة مزدوجة</td></tr>
<tr><td colspan="2">تعيين إلى الكتل الغير مباشرة (4 / $block_size$) (1024 في حالة كتل 4 كيلوبايت)</td><td rowspan="2">من 0 إلى (4 / $block_size$)</td></tr>
<tr><td>تشير إلى</td><td>إزاحة كتلة غير مباشرة</td></tr>
<tr><td colspan="2">تعيين إلى الكتل (4 / $block_size$) (1024 في حالة كتل 4 كيلوبايت)</td><td>من 0 إلى (4 / $block_size$)</td></tr>
</table>

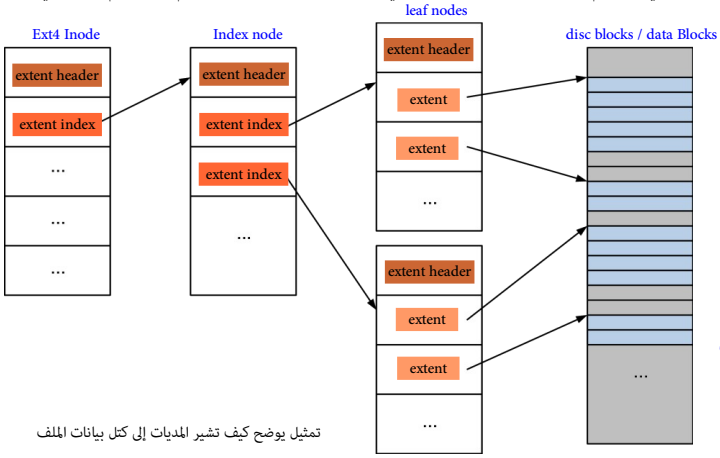
* في هذه الصيغة القسمة كانت على 4 لأن كل رقم كتلة منطقية مخزن في 4 بايت.

لاحظ مع مخطط ربط الكتل هذا، سيكون من الضروري ملء الكثير من بيانات الربط [122] حتى مع الملف الكبير المتماس! وهذا هو السبب في إنشاء مخطط ربط أو تعيين المدبات [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]



في ext4، ربط الكتل المنطقية إلى الملف استبدل بشجرة مديات [120][97][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 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 السانات الوصفية وتحسن في فاعلية القرص. لكن الميزة، تستلزم تعيين علم المديات في Inode.



تمثيل يوضح 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 منها 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 تحسدات 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 تحسدات 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 في شجرة extent tree مسجلة في بنية struct ext4_extent_header، (بطول 12 بايت)

```
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 00 00 |........_. .....|
```

رمز تذكري	إزاحة	نوع	حجم		
eh_magic	0x00 (00)	__le16	2	Magic number, 0xF30A	رقم سحري 0xF30A
eh_entries	0x02 (02)	__le16	2		عدد مدخلات المدى الصالحة التي بعد الترويسة
eh_max	0x04 (04)	__le16	2		العدد الأقصى للمدخلات المدى بعد الترويسة
eh_depth	0x06 (06)	__le16	2	4*(((blocksize - 12)/12)^n) >= 2^32 is 5	عمق عقدة المدى هذه في شجرة المديات [101]
eh_generation	0x08 (08)	__le32	4		رقم توليد الشجرة (هذا يستخدمه نظام الملفات الموزع المتوازي، لوستر Lustre، وليس النظام المعياري ext4)
رمز تذكري	إزاحة	نوع	حجم		مدخلة المدى من أجل العقد الداخلية في شجرة المديات، تعرف أيضا باسم index nodes، ومسجلة في بنية struct ext4_extent_idx، (بطول 12 بايت)
ei_block	0x00 (00)	__le32	4	رقم الكتلة المنطقية	عقدة الفهرسة index node هذه تغطي كتل الملف من الكتلة 'block' فصاعدا (أين نجد المديات تحت هذه العقدة في الشجرة)
ei_leaf_lo	0x04 (04)	__le32	4	عنوان الكتلة الفيزيائية	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
ei_leaf_hi	0x08 (08)	__le16	2		16-بت العليا من الحقل السابق
ei_unused	0x0A (10)	__le16	2		غير مستخدم !

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

```
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 |................|
```

رمز تذكري	إزاحة	نوع	حجم		
ee_block	0x00 (00)	__le32	4	رقم الكتلة المنطقية	رقم كتلة الملف الأولى التي يغطيها هذا المدى (مكان هذا المدى النسبي إلى بداية الملف) [115]
ee_len	0x04 (04)	__le16	2		عدد الكتل التي يغطيها هذا المدى [102]
ee_start_hi	0x06 (06)	__le16	2	رقم الكتلة الفيزيائية من أول كتلة في المدى، أي البداية الفعلية للمدى على القرص	رقم الكتلة (16-بت علىا) التي يشير لها هذا المدى
ee_start_lo	0x08 (08)	__le32	4	عنوان الكتلة الفيزيائية	رقم الكتلة (32-بت المنخفضة) التي يشير لها هذا المدى

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

رمز تذكري	إزاحة	نوع	حجم		بنية struct ext4_extent_tail (بطول 4 بايت)
eb_checksum	0x00 (00)	__le32	4	crc32c(uuid+inum+igeneration+extentblock)	تدقيق مجموع كتلة المدى extent block

البيانات المضمنة / البيانات المباشرة — Inline Data

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

مدخلات الدليل — Directory Entries

الأدلة تستخدم من أجل تنظيم الملفات بشكل مرتب/ 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 في بنية Inode.

المدخلة الثانية في جدول Inode table تتضمن Inode يشير إلى بيانات الدليل الجذر Root directory ؛ المحدد بالثابت EXT4_ROOT_INO. في أنظمة يونكس وشبه يونكس (مثل لينكس، ومينيكس) 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 مدخلة دليل شاغرة لتجاوز فهارس الهاش hash indexes.

في نظام ملفات 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 بالروابط 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) من أجل مدخلة 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 لا تجزأ 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 تمتلك 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 struct ext4_dir_entry في حالة تعطيل 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، (النسخة الأصلية)			
رقم inode الذي تشير إليه مدخلة الدليل هذه	4	__le32	0x00 (00)
طول مدخلة الدليل هذه	2	__le16	0x04 (04)
طول اسم الملف	2	__le16	0x06 (06)
اسم الملف	255	char	0x08 (08)

بما أن أسماء الملفات لن تكون أطول من 255 بايت، صيغة مدخلة الدليل (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 (النسخة الحديثة)

```
debugfs -R "cat <518977>" /dev/sda1 | hexdump -C
     00 01 0 203 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 0e 00 02 01 69 73 00 00 |this........is..|
0030 cd ee 07 00 0c 00 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 |................|
1000
```

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

File type code :

0x00	EXT4_FT_UNKNOWN	Unknown	نوع ملف مجهول
0x01	EXT4_FT_REG_FILE	Regular file	ملف اعتيادي
0x02	EXT4_FT_DIR	Directory file (Directory)	ملف دليل
0x03	EXT4_FT_CHRDEV	character special file (character device)	ملف جهاز محرفي
0x04	EXT4_FT_BLKDEV	block special file (block device)	ملف جهاز كتل
0x05	EXT4_FT_FIFO	Named pipe / FIFO ! (Buffer File)	ملف صوان / أنبوبة اتصال مسماة
0x06	EXT4_FT_SOCK	Unix domain socket	ملف مقبس (IPC socket)
0x07	EXT4_FT_SYMLINK	Symbolic link, soft link, symlink	وصلة رمزية (وصلة لينة) [111]

لإضافة تدقيق المجموع إلى كتل الدليل الكلاسيكية هذه، وضعت المدخلة الزائفة struct ext4_dir_entry في نهاية كل كتلة طرفية leaf block لحفظ تدقيق المجموع. وسيكون طول مدخلة الدليل 12 بايت. مع تعيين رقم مؤشر الفهرسة و name_len إلى الصفر لخداع البرمجيات القديمة وجعلها تتجاهل مدخلة الدليل التي تبدو شاغرة، وتدقيق المجموع سيكون في مكان الاسم الاعتيادي.

بنية ذيل مدخلة الدليل struct ext4_dir_entry_tail		حجم / نوع		إزاحة	رمز تذكري
رقم inode (يجب أن يكون صفر)	Pretend to be unused	4	__le32	0x00 (00)	det_reserved_zero1
طول مدخلة الدليل هذه، (يجب أن يكون 12)	12	2	__le16	0x04 (04)	det_rec_len
طول اسم الملف، (يجب أن يكون صفر)	Zero name length	1	__u8	0x06 (06)	det_reserved_zero2
نوع الملف (زائف) (يجب أن يكون 0xDE)	0xDE, fake file type	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

تدقيق مجموع الكتلة الطرفية للدليل سيكون بحساب : FS UUID، ورقم inode للدليل، ورقم توليد inode للدليل، وكامل كتلة مدخلة الدليل إلى مدخلة الدليل المزيفة fake (بدون حسابها)

أدلة شجرة هاش

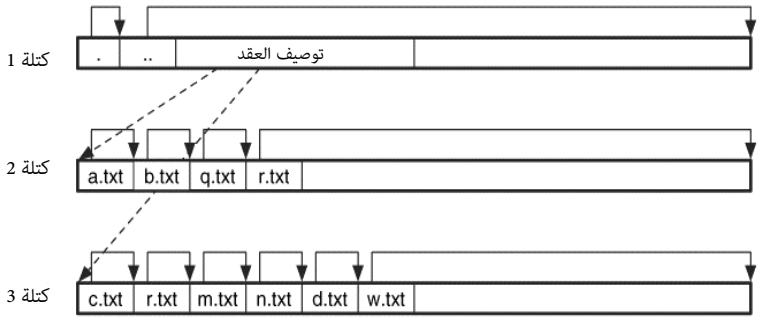
Hash Tree Directories

المصفوفة الخطية للمدخلات الدليل لم تكن ذات فاعلية لذا 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 EXT4_INDEX_FL في Inode، الدليل المعني سوف يستخدم شجرة HTree (الدليل المفهرس) [42] في تنظيم وإيجاد مدخلات الدليل. للتوافق 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 توصيف العقد node descriptors في تحديد الكتلة التي يقفز إليها من أجل قيمة الهاش. في الخطاطة التالية تظهر عدة ملفات في طرفيتين. أول كتلة تتضمن تروبسة وتوصيف عقد، والثانية والثالثة تتضمن مدخلات دليل ملفات.



الدليل مع شجرات الهاش hash trees وطرفيتين two leaves. الشجرة تستخدم مدخلات الدليل 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 تروبسة، تبدأ بعد مدخلة الدليل ".." حقول تروبسة توصيف العقد node descriptor header تظهر في الجدول أسفل.

إذا كان حقل dx_root.info.indirect_levels بقيمة غير الصفر شجرة HTree سوف تمتلك 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 مصفرة متبوعة بالربط بين قيم الهاش الدنيا وعناوين الكتل minor_hash->block map لإيجاد العقد الطرفية leaf nodes. العقد الطرفية تتضمن مصفوفة خطية linear array (أي linked list) من جميع مدخلات الدليل 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، (أي الكتلة) 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 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 الثانية لإيجاد رقم الكتلة الثالثة المقابل. رقم الكتلة الثالثة هذه ستكون مصفوفة خطية من مدخلات الدليل. وللبحث المتعمق في الدليل كمصفوفة خطية (كما تفعل الشفرة القديم)، الشفرة ببساطة تقرأ كل كتلة بيانات في الدليل. الكتل المستخدمة من أجل HTree ستبدو بدون مدخلات (بجانب '.' و '..') ولذلك فقط العقد الطرفية leaf nodes ستبدو بمضمون مهم.

في حالة تمكين تدقيق مجاميع البيانات الوصفية، 8 بايت الأخيرة من كتلة الدليل (بطول واحدة 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 موجود مساحة من أجل تدقيق المجموع)

تدقيق المجموع سيكون بحساب: معرف FS UUID، وتروبسة فهرس HTree index (dx_root أو dx_node) وجميع HTree_indices (dx_entry) المستخدمة، وكتلة الذيل (dx_tail)

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

```
debugfs -R "cat <123456>" /dev/sda1 | hexdump -C
      00 01 0 203 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 inode لهذا الدليل [130]	this directory "."	struct ext4_dir_entry_2 (fake_dirent 1) مدخلة دليل قائمة متصلة (للتوافق خلفيا) [117] [99]	حجم / نوع	إزاحة	رمز تذكري
طول هذه التسجيلة، 12			4 __le32	0x00 (00)	dot.inode
طول هذا الاسم، 1			2 __le16	0x04 (04)	dot.rec_len
نوع ملف هذه المدخلة (0x02 تعني دليل) (في حالة تعيين علم الميزة)	EXT2_FT_DIR=2		1 u8	0x06 (06)	dot.name_len
".\0\0\0"			1 u8	0x07 (07)	dot.file_type
			4 char	0x08 (08)	dot.name[4]
رقم inode الدليل الأم	parent directory ".."	struct ext4_dir_entry_2 (fake_dirent 2) مدخلة دليل قائمة متصلة (للتوافق خلفيا)	4 __le32	0x0C (12)	dotdot.inode
12 – block_size، طول التسجيلة يكفي لتغطية جميع بيانات Htree			2 __le16	0x10 (16)	dotdot.rec_len
طول الاسم، 2			1 u8	0x12 (18)	dotdot.name_len
نوع ملف هذه المدخلة (0x02 تعني دليل) (في حالة تعيين علم الميزة)	EXT2_FT_DIR=2		1 u8	0x13 (19)	dotdot.file_type
"..\0\0"			4 char	0x14 (20)	dotdot_name[4]

صفر				dx_root_info معلومات جذر الدليل المفهرس ترويسة توصيف عقدة شجرة الهاش the 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 !	Legacy	DX_HASH_LEGACY	0x00				
نصف دالة الهاش التشفيرية إم دي 4	Half MD4	DX_HASH_HALF_MD4	0x01				
خوارزمية التشفيرية الصغرة !	Tea	DX_HASH_TEA	0x02				
تراثي!، (عدد صحيح) لا يحمل إشارة	Legacy, unsigned	DX_HASH_LEGACY_UNSIGNED	0x03				
نصف دالة الهاش التشفيرية إم دي 4، لا يحمل إشارة	Half MD4, unsigned	DX_HASH_HALF_MD4_UNSIGNED	0x04				
خوارزمية التشفيرية الصغرة!، لا يحمل إشارة	Tea, unsigned	DX_HASH_TEA_UNSIGNED	0x05				
طول معلومات الشجرة، هذه 8 بت تمثل طول بنية معلومات الدليل المفهرس (dx_root) ؛ حاليا تساوي 0x08					1 u8	0x1D (29)	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]					1 u8	0x1E (30)	struct dx_root_info.indirect_levels
محجوز					1 u8	0x1F (31)	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 the first node descriptor entry (حالة خاصة) [118] [43]	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				dx_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		المدخلة المخفية / المزيفة!	4 __le32	0x00 (00)	fake.inode
حجم الكتلة، من أجل إخفاء جميع بيانات dx_node			2 __le16	0x04 (04)	fake.rec_len
صفر، لا يوجد اسم لمدخلة الدليل هذه الغير مستخدمة "unused"_directory entry			1 u8	0x06 (06)	name_len
صفر، لا يوجد نوع ملف لمدخلة الدليل هذه الغير مستخدمة "unused"_ directory entry			1 u8	0x07 (07)	file_type
العدد الأقصى من مدخلات الدليل المفهرس dx_entries التي يمكن أن تتبع هذه الترويسة، زائد 1 من أجل الترويسة نفسها			2 __le16	0x08 (08)	limit
العدد الفعلي من مدخلات الدليل المفهرس dx_entries التي تتبع هذه الترويسة، زائد 1 من أجل الترويسة نفسها			2 __le16	0x0A (10)	count
رقم الكتلة (داخل ملف الدليل) الذي يقابل قيمة هاش الأدنى من هذه الكتلة. هذه القيمة مخزنة في الكتلة الأم			4 __le32	0x0E (14)	Block
أكبر عدد ممكن من مدخلات struct dx_entry ـ قيم 8-بت يمكن أن يتناسب مع بقية حجم كتلة البيانات			struct dx_entry --	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 the hash tree node descriptor entries	4 __le32	0x00 (00)	Hash
رقم كتلة العقدة التالية في HTree (داخل ملف الدليل، وليس كتل نظام الملفات) Block number			4 __le32	0x04 (04)	Block

بنية تدقيق المجموع dx_tail (بطول 8 بايت) ستكون في نهاية كل كتلة htree block			حجم / نوع	إزاحة	رمز تذكري
محجوز			4 u32	0x00 (00)	dt_reserved
تدقيق مجموع كتلة دليل HTree		crc32c(uuid+inum+dxblock)	4 __le32	0x04 (04)	dt_checksum

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] (للملف الاعتيادي والدليل) هذه الخصائص الممتدة هي امتدادات للخصائص العادية التي ترتبط بكافة inodes في النظام. وتستخدم غالبا في توفير تأدية وظيفية إضافية في نظام الملفات – مثلا ميزّات أمان إضافية مثل قوائم التحكم بالنفاذ ACLs التي يمكن تطبيقها كخصائص ممتدة. وهذه سيكون النفاذ إليها بصفتها كائنات ذرية تحلب [46] القراءة كامل قيمة الخاصية وتخزنها في الصوان. والكتابة تستبدل أية قيمة سابقة بقيمة جديدة.

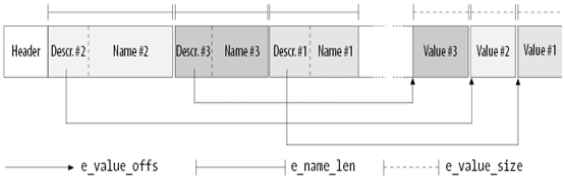
الخصائص الممتدة (xattrs) تخزن في كتلة بيانات مستقلة على القرص تشير إليها حقول inode.i_file_acl* من inodes ويبدو أن أول استخدام للخصائص الممتدة كان من أجل تخزين أذون ملفات ACLs وبيانات الأمن الأخرى (selinux).أيضا يمكن للمستخدمين مع خيار الوصل user_xattr تخزين الخصائص الممتدة طالما أن جميع أسماء الخصائص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

الخصائص 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 - (28 + 128) = 100 بايت متوفرة لتخزين الخصائص الممتدة في inode. المكان الثاني الذي يمكن أن نجد فيه الخصائص الممتدة هو الكتلة التي تشير لها حقول inode.i_file_acl.لكن منذ إصدار نواة لينكس 3.11، لا يمكن لهذه الكتلة أن تتضمن مؤشر إلى كتلة خصائص ممتدة ثانية (أو حتى بقية كتل العنقود) نظريا، يمكن لكل قيمة خاصة أن تخزن في كتلة بيانات مستقلة، رغم أن الشفرة لا تسمح بذلك منذ إصدار نواة لينكس 3.11.

عموما، المفاتيح يفترض أن تكون سلاسل ASCIIZ، بينما القيم يمكن أن تكون سلاسل أو بيانات ثنائية.

الخصائص الممتدة، عندما تخزن بعد inode تملك هذه التروسة ext4_xattr_ibody_header (بطول 4 بايت)		حجم / نوع		إزاحة	رمز تذكري
رقم سحري من أجل التعريف، (مشغل لينكس بعين. هذه القيمة، لكن، يبدو أن e2fsprogs لا تتفحص هذه القيمة ؟)	0xEA020000	4	__le32	0x00 (00)	H_magic
بداية كتلة الخصائص الممتدة struct ext4_xattr_header، (بطول 32 بايت)		حجم / نوع		إزاحة	رمز تذكري
رقم سحري من أجل التعريف (32 بت)	EXT4_XATTR_MAGIC = 0xEA020000	4	__le32	0x00 (00)	h_magic
تَعْداد المراجع [38] (عدد الملفات التي تستخدم هذه الكتلة / التي تملك نفس الخصائص الممتدة)		4	__le32	0x04 (04)	h_refcount
عدد كتل القرص المستخدمة من قبل الخصائص الممتدة [44]		4	__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)		4	__le32	0x0C (12)	h_hash
تدقيق مجموع كتلة الخصائص الممتدة (FS UUID، رقم كتلة 64-بت لكتلة الخصائص الممتدة، وكامل الكتلة (التروسة+ المدخلات))	crc32c(uuid+id+xattrs)	12	__le32	0x10 (16)	h_checksum
		4	__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 وسيكون e_name_index, ثم e_name_len وأخيرا e_name الخصائص المخزنة داخل Inode لا تحتاج أن تكون مخزنة بترتيب مفروز.



تخطيط كتلة تتضمن خصائص ممتدة

بنية مدخلة الخصائص (مدخلات أسماء الخصائص الممتدة) struct ext4_xattr_entry؛			حجم / نوع		إزاحة	رمز تذكري
طول الاسم (8 بت لا تحمل إشارة) (طول الاسم يحدد طول المدخلة، والمدخلة التالية تبدأ عند حدود 4 بايت التالية)			1	__u8	0x00 (00)	e_name_len
فهرس اسم الخاصية [110] (8 بت لا تحمل إشارة) هذه يمكن أن تكون إحدى القيم التالية:	Attribute name index :		1	__u8	0x01 (01)	e_name_index
	مؤشر الاسم	سابقة المفتاح (مساحات الأسماء / مجالات أسماء)	ملاحظة			
	0	NULL	لا توجد سابقة			
	1	"user." (User space attribute)	لا يملك قيود بخصوص التسمية أو المحتويات			
	2	"system.posix_acl_access"	تستخدمها النواة من أجل ACLs			
	3	"system.posix_acl_default" (directories only)				
	4	"trusted." (Trusted space attribute)	تستخدمها فقط النواة !			
	5	LUSTRE	غير مستخدم في لينكس!			
	6	"security." (Security space attribute)	تستخدمها SELinux			
	7	"system."	فقط inline_data			
	8	"system.richacl"	فقط أنوية SuSE			
قيمة الإزاحة (16 بت لا تحمل إشارة) موقع قيمة هذه الخاصية على كتلة القرص حيث تخزن.(أي إزاحة البايت في الكتلة المحددة) [108]			2	__le16	0x02 (02)	e_value_offs
inode الذي يخزن القيمة. رقم 0 يشير إلى أن القيمة في نفس الكتلة مثل هذه المدخلة. هذا الحقل يستخدم فقط في حالة تمكين INCOMPAT_EA_INODE			4	__le32	0x04 (04)	e_value_inum
حجم قيمة الخاصية (32 بت لا تحمل إشارة) (عدد بايتات في القيمة) size of attribute value			4	__le32	0x08 (08)	e_value_size
قيمة الهاش [31] الخاصة بقيمة الخاصة واسم الخاصة [109] hash value of name and value			4	__le32	0x0C (12)	e_hash
اسم الخاصية attribute name. لا يشمل المحرف الصفري (رمز لاشيء) في الذيل ! trailing NULL			16	char	0x10 (16)	e_name[e_name_len]

قيم الخصائص Attribute values يمكنها أن تتبع نهاية جدول المدخلات ويبدو أن هناك شرطًا بأن تكون المحاذاة على حدود 4 بايت (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 تعيين إلى الصفر للدلالة على نهاية لائحة المفاتيح key list

كتلة الخصائص الممتدة: تملك ثلاثة أقسام. أول 32 بت تستخدمها الترويسة header، بعد الترويسة القسم الثاني يتضمن لائحة بمدخلات أسماء الخصائص. القسم الثالث يبدأ من نهاية الكتلة ويتجه للأعلى. ويتضمن القيم لكل زوج خصائص، التي قد لا تكون في نفس الترتيب مثل مدخلات الأسماء. يمكن رؤية ذلك في الشكل أعلاه. بالمناسبة يمكن طرح جميع الخصائص الممتدة للملف بواسطة الأمر getfattr -d -m - file

فهارس أسماء الخصائص — Attribute Name Indices

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

مؤشر الاسم	سابقة المفتاح (مساحات الأسماء)	ملاحظة
0	NULL	لا توجد سابقة
1	"user."	من أجل تسجيل الخصائص properties المحددة من التطبيقات applications
2	"system.posix_acl_access"	تستخدمها النواة من أجل ACLs
3	"system.posix_acl_default"	
4	"trusted."	من أجل تسجيل خصائص ينبغي للنواة فقط النفاذ إليها
6	"security."	من أجل تسجيل الخصائص الأمنية للملف / تستخدمها SELinux
7	"system."	من أجل تسجيل الخصائص الأخرى المرتبطة بالنظام (فقط inline_data)
8	"system.richacl"	فقط أنوية SuSE

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

قوائم التحكم بالنفاذ (معيار بوزيكس) — POSIX ACLs

قوائم التحكم بالنفاذ [121] معيار بوزيكس POSIX ACLs 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 أذون ACLs للمستخدم والمجموعة المسماة named user, named group. (بيانات الجدول التالي من ملف e2fsprogs-1.44.0/lib/ext2fs/ext4_acl.h.).

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

بنية بيانات مدخلة posix_acl_xattr_entry					نوع	إزاحة	رمز تذكري
حقل النوع (وسم) (tag) في مدخلة ACL يحدد نوع الإذن الذي من أجله كانت المدخلة.:					__le16	0x00 (00)	e_tag
	0x00			ACL_UNDEFINED_TAG			
المستخدم المحدد في inode (مدخلة تعلن عن حقوق النفاذ للمالك الملف)	0x01	Owner	user::rwx	ACL_USER_OBJ			
المجموعة (المستخدمون) المحددة في inode (مدخلة تعلن عن حقوق النفاذ للمجموعة الملف)	0x04	Owning Group	group::rwx	ACL_GROUP_OBJ			
جميع المستخدمون الآخرون other users (مدخلة تعلن عن حقوق النفاذ للعمليات التي لا تطابق أية مدخلة أخرى في ACL	0x20	Other	other::rwx	ACL_OTHER			
قناع الحقوق النافذة ! (الفعالة) Effective rights mask (مدخلة تعلن عن حقوق النفاذ الأقصى الممكن منحوها حسب النوع)	0x10	Mask	mask::rwx	ACL_MASK			
المستخدم المحدد في attribute (مدخلة تعلن عن حقوق النفاذ للمستخدمين حددهم مصنف ! المدخلة)	0x02	Named User	user:name:rwx	ACL_USER			
المجموعة (المستخدمون) المحدد في attribute (مدخلة تعلن عن حقوق النفاذ للمجموعات حددهم مصنف ! المدخلة)	0x08	Named Group	group:name:rwx	ACL_GROUP			
أذون النفاذ					__le16	0x02 (02)	e_perm
تنفيذ / بحث	0x01	Execute					
كتابة	0x02	Write					
قراءة	0x04	Read					
هوية المستخدم / المجموعة User / Group ID (ليس مضمن في بعض الأنواع)					__le32	0x04 (04)	e_id

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

هذه البنية ستكتب إلى رقم الكتلة المحفوظ في حقل 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.

قيد الحوادث (أو سِجلّ الحوادث) journal[113] ميزة ظهرت في ext3 للحماية البيانات من التلف في حال انهيار النظام، هناك خمسة أنواع من الكتل يمكن أن يملكها قيد الحوادث، الأربعة الأولى منها تنفيذية وتعرف بـ: Super block, Descriptor, Commit, Revoke. النوع الخامس يخزن البيانات الوصفية أو البيانات التي تم تسجيلها في قيد الحوادث 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. النوع الخامس يمكن أن يكون كتل بيانات وصفية تحتفظ بنسخ من Inodes تم تعديلها في نظام الملفات، إذا كان قيد الحوادث يستخدم نمط ordered أو write back وزائد كتل المحتوى إن كان قيد الحوادث في نمط journaled. الهياكل الداخلية للأنواع المختلفة من الكتل ملخصة في الجداول التالية.

ext4 يخصص منطقة متواصلة صغيرة على القرص (128 ميغابايت) داخل نظام الملفات كموضع سريع للكتابة البيانات "المهمة" على القرص. بمجرد كتابة كامل إجراء البيانات [84] الهامة إلى القرص وتخليصها [123] (flushed) من ذاكرة التخزين المؤقت للكتابة القرص [124] disk write cache، يتم كتابة سجل [22] بالبيانات المنفذة إلى قيد الحوادث. في وقت لاحق، شفرة قيد الحوادث ستكتب الإحراءات إلى موقعهم النهائي على القرص (قد ينطوي هذا على الكثير من السعي "أي تحريك رأس القراءة/الكتابة" أو كثير 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 قيد الحوادث بالكامل حتى آخر سجل تنفيذ لضمان ذرية [46]كل ما يكتب من خلال قيد الحوادث إلى القرص. هذا سيضمن عدم توقف نظام الملفات في منتصف الطريق عند تحديث البيانات الوصفية. لأسباب تتعلق بالأداء، ext4 يكتب فقط البيانات الوصفية للنظام الملفات من خلال قيد الحوادث. 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 (راجع أكثر دليل أداة mount في لينكس).

وإن كان النمط هو data=journal، جميع البيانات والبيانات الوصفية تكتب إلى القرص من خلال قيد الحوادث. هذا الخيار أبطأ لكنه الأكثر أمانا.

أما إن كان النمط هو data=writeback، لا يتم تخليص [123](flushed) كتل البيانات الملوثة (معدلة!) [26] dirty data blocks إلى القرص حتى يتم كتابة البيانات الوصفية إلى القرص من خلال قيد الحوادث.

مؤشر فهرسة قيد الحوادث journal inode عادة سيكون رقم 8. وأول 68 بايت من journal inode مكررة في الكتلة العليا[112] ext4 superblock. قيد الحوادث نفسه عبارة عن ملف اعتيادي (لكنه مخفي) داخل نظام الملفات. الملف عادة يستهلك مساحة مجموعة كتل كاملة (أي 128 ميغابايت)، رغم أن mke2fs يحاول وضعه في منتصف القرص.

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.

ملاحظة: نظام ملفات ocfs2_ و ext4 كلاهما يستخدم طبقة jbd2 [53]. حجم قيد الحوادث الأقصى المضمن في ext4 هو 2^{32} كتلة. ويبدو أن Jbd2 نفسه لا يكترث لذلك.

تخطيط قيد الحوادث

Layout

عموما، قيد الحوادث [113] سيكون بالصيغة التالية:

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

إجراء واحد One transaction

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

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

إجراء واحد One transaction

ترويسة الكتلة

Block Header

كل كتلة في قيد الحوادث تبدأ بترويسة 12-بايت struct journal_header_s (هذه الترويسة معيارية لجميع 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.............|
```

حجم	نوع	إزاحة	رمز تذكري			
4	__be32	0x00 (00)	h_magic	0xC03B3998		رقم سحري من أجل jbd2
4	__be32	0x04 (04)	h_blocktype	Descriptor block types :		وصف مضمون هذه الكتلة، (نوع كتلة التوصيف) سيكون إحدى هذه :
				1	JBD2_DESCRIPTOR_BLOCK	Descriptor توصيف.هذه الكتلة تسبق سلسلة من كتل البيانات التي كتبت من خلال قيد الحوادث أثناء الإجراء.
				2	JBD2_COMMIT_BLOCK	Commit سجل تنفيذ الكتلة.هذه الكتلة تدل على اكتمال الإجراء.
				3	JBD2_SUPERBLOCK_V1	Journal superblock, v1 الكتلة العليا في قيد الحوادث، النسخة 1
				4	JBD2_SUPERBLOCK_V2	Journal superblock, v2 الكتلة العليا في قيد الحوادث، النسخة 2
				5	JBD2_REVOKE_BLOCK	Revoke تسجيلات إلغاء الكتلة. هذه تسرع الاسترداد بتمكين Journal من تخطي كتابة الكتل التي يعاد كتبتها لاحقا
4	__be32	0x08 (08)	h_sequence	هوية الإجراء المصاحبة لهذه الكتلة.		

الكتلة العليا في قيد الحوادث أبسط بكثير مقارنة مع ext4 super block[112]. البيانات المهمة المحفوظ في الكتلة هي حجم journal، ومكان بداية سجل الاجراءات.(جميع الحقول بترتيب بايت نهوي كبير)

كتلة journal superblock مسجلة بالشكل 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 0e 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 00 |................|
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]
03f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
0400
```

رمز تذكري	إزاحة	حجم / نوع			
s_header	0x00 (00)	journal_header_t (بايت 12)		journal_header_t s_header	ترويسة عامة، من أجل التعريف بالكتلة العليا superblock
معلومات ثابتة تصف قيد الحوادث journal					
s_blocksize	0x0C (12)	__be32	4	journal device blocksize	حجم كتلة جهاز قيد الحوادث.
s_maxlen	0x10 (16)	__be32	4	total blocks in journal file	عدد الكتل الإجمالي في قيد الحوادث
s_first	0x14 (20)	__be32	4	first block of log information	أول كتلة من معلومات السجل.
معلومات ديناميكية تصف الوضعية الحالية للسجل					
s_sequence	0x18 (24)	__be32	4	first commit ID expected in log	أول هوية تنفيذ متوقع في السجل
s_start	0x1C (28)	__be32	4	blocknr of start of log	رقم كتلة بداية السجل (الصفر في هذا الحقل لا يعني أن قيد الحوادث نظيف !)
s_errno	0x20 (32)	__be32	4	Error value, as set by journal_abort()	قيمة الخطأ، كما تم تعيينها بواسطة jbd2_journal_abort ()
بقية الحقول صالحة فقط في النسخة 2 من superblock.					
s_feature_compat;	0x24 (36)	__be32	4	Compatible feature set :	مجموعة أعلام الميزّات المتوافقة [27]
				0x01 JBD2_FEATURE_COMPAT_CHECKSUM	قيد الحوادث يحفظ تدقيق المجاميع على كتل البيانات
s_feature_incompat	0x28 (40)	__be32	4	Incompatible feature set :	مجموعة أعلام الميزّات الغير متوافقة
				0x01 JBD2_FEATURE_INCOMPAT_REVOKE	قيد الحوادث يملك تسجيلات إلغاء الكتلة block revocation records
				0x02 JBD2_FEATURE_INCOMPAT_64BIT	قيد الحوادث يستطيع التعامل مع أرقام / أعداد كتلة 64-بت
				0x04 JBD2_FEATURE_INCOMPAT_ASYNC_COMMIT	تنفيذ قيد الحوادث لا تزامني
				0x08 JBD2_FEATURE_INCOMPAT_CSUM_V2	قيد الحوادث هذا يستخدم النسخة v2 من صيغة تدقيق المجموع. على القرص [93]
				0x10 JBD2_FEATURE_INCOMPAT_CSUM_V3	قيد الحوادث هذا يستخدم النسخة v3 من صيغة تدقيق المجموع على القرص. [94]
s_feature_ro_compat	0x2C (44)	__be32	4	Read-only compatible feature set :	مجموعة أعلام الميزّات المتوافقة- في وضعية للقراءة فقط [27] حاليا هذه لا توجد.
				-- --	--
s_uuid[16]	0x30 (48)	__u8	16	معرف UUID ـ 128-بت من أجل قيد الحوادث. هذا يقارن بالنسخة في ext4 super block في زمن وصل نظام الملفات.	
s_nr_users	0x40 (64)	__be32	4	عدد أنظمة الملفات التي تتشارك قيد الحوادث هذا.	
s_dynsuper	0x44 (68)	__be32	4	موقع نسخة الكتلة العليا الديناميكية dynamic super block (غير مستخدم؟)	
s_max_transaction	0x48 (72)	__be32	4	حد عدد كتل قيد الحوادث لكل إجراء. (غير مستخدم؟) Limit of journal blocks per transaction	
s_max_trans_data	0x4C (76)	__be32	4	حد عدد كتل البيانات لكل إجراء. (غير مستخدم؟) Limit of data blocks per transaction	
s_checksum_type	0x50 (80)	__u8	1	نوع خوارزمية تدقيق المجموع المستخدمة من أجل قيد الحوادث، الأرجح ستكون 1 أو 4	Checksum algorithm :
				1 JBD2_CRC32_CHKSUM	crc32
				2 JBD2_MD5_CHKSUM	md5
				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 التي تصف المواقع النهائية للكتل البيانات التي تتبع في قيد الحوادث.كتل التوصيف ستكون open-coded [35] بدل وصفها كاملا بواسطة بنية بيانات،

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

```
dd if=journal.dd bs=4096 skip=2 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 01  00 0c d5 cd 00 00 00 00  |.;9.............|
0010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0020  00 00 00 00 00 00 08 04  ed 00 00 00 02 00 08 04  |................`|
0030  00 00 00 02 00 08 01 5b  00 00 00 02 00 18 00 23  |.......[.......#|
[REMOVED]
0530  00 00 00 0a 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0540  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
1000
```

	حجم / نوع		إزاحة	رمز تذكري
ترويسة كتلة عامة	12	journal_header_t	0x00 (00)	(open coded)
أوسمة tags تكفي لملء الكتلة أو تكفي لوصف جميع كتل البيانات التي تتبع كتلة التوصيف هذه	--	struct journal_block_tag_s	0x0C (12)	open coded array[]

وسم الكتلة block tag يستخدم في وصف صوان buffer واحد في journal. أوسمة كتلة قيد الحوادث Journal block tags ستملك إحدى الصيغ التالية، بناء على ميزة قيد الحوادث وأعلام وسم الكتلة block tag التي تم تعيينها.

			حجم / نوع	إزاحة	رمز تذكري
عند تعيين JBD2_FEATURE_INCOMPAT_CSUM_V3، وسم كتلة قيد الحوادث سيكون struct journal_block_tag3_s (بحجم 16 أو 32 بايت)					
رقم الكتلة على القرص / الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص. (32-بت المنخفضة)			4	0x00 (00)	t_blocknr
الأعلام التي تأتي مع التوصيف، ستكون أيا من هذه:		the journal tag flags word :	4	0x04 (04)	t_flags
تخطي الكتلة على القرص. في حالة تطابق أول 4 بايت من كتلة البيانات مع الرقم السحري في jbd2	0x01	JBD2_FLAG_ESCAPE			
هذه الكتلة تملك نفس معرف UUID،مثل السابقة، ولذلك، تم إسقاط حقل UUID.	0x02	JBD2_FLAG_SAME_UUID			
كتلة البيانات حذفها هذا الإجراء (غير مستخدم؟)	0x04	JBD2_FLAG_DELETED			
هذا أخر وسم tag في كتلة التوصيف هذه	0x08	JBD2_FLAG_LAST_TAG			
الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص (32-بت العليا) هذه صفر في حالة تعطيل JBD2_FEATURE_INCOMPAT_64BIT			4	0x08 (08)	t_blocknr_high
تدقيق مجموع كل من معرف قيد الحوادث UUID، ورقم المتتالية، وكتلة البيانات.		crc32c(uuid+seq+block)	4	0x0C (12)	t_checksum
هذا الحقل يبدو بترميز مفتوح [35] open coded. ويأتي دائما في نهاية الوسم. بعد t_checksum. هذا الحقل لن يكون موجود في حالة تعيين علم "SAME_UUID"					
UUID الذي يأتي مع هذا الوسم.هذا الحقل يبدو منسوخ من حقل j_uuid في struct journal_s، لكن فقط tune2fs يمكنها التعامل مع ذلك الحقل			16	0x08 / 0x0C	Uuid[16]
عند تعطيل JBD2_FEATURE_INCOMPAT_CSUM_V3 وسم كتلة قيد الحوادث يكون struct journal_block_tag_s، (بحجم 8، 12، 24 أو 28 بايت)					
رقم الكتلة على القرص / الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص.(32-بت المنخفضة)			4	0x00 (00)	t_blocknr
تدقيق مجموع: معرف قيد الحوادث UUID، ورقم المتتالية وكتلة البيانات لاحظ: تخزن فقط 16 بايت المنخفضة		truncated crc32c(uuid+seq+block)	2	0x04 (04)	t_checksum
الأعلام التي تأتي مع الوصف. أيا من هذه:		the journal tag flags word :	2	0x06 (06)	t_flags
تخطي الكتلة على القرص، فقط في حالة تطابق أول 4 بايت من كتلة البيانات مع الرقم السحري في jbd2	0x01	JBD2_FLAG_ESCAPE			
هذه الكتلة، تملك نفس معرف UUID، مثل السابقة ولذلك تم إسقاط حقل UUID.	0x02	JBD2_FLAG_SAME_UUID			
كتلة البيانات حذفها إجراء. (غير مستخدم؟)	0x04	JBD2_FLAG_DELETED			
هذا أخر وسم في كتلة التوصيف هذه.	0x08	JBD2_FLAG_LAST_TAG			
الحقل التالي سيكون موجود فقط في حالة كانت الكتلة العليا super block تشير إلى دعم أرقام / أعداد كتلة 64-بت.					
الموقع حيث يجب أن تنتهي كتلة البيانات المقابلة على القرص. (32-بت العليا)			4	0x08 (08)	t_blocknr_high
هذا الحقل يبدو مفتوح الترميز open coded. ويأتي دائما في نهاية الوسم. بعد t_flags أو t_blocknr_high.هذا الحقل لن يكون موجود في حالة تعيين علم نفس المعرف "SAME_UUID"					
UUID الذي يأتي مع هذا الوسم.هذا الحقل يبدو منسوخ من حقل j_uuid في struct journal_s، لكن فقط tune2fs تستطيع التعامل مع ذلك الحقل.			16	0x08 / 0x0C	Uuid[16]
ذيل كتلة revoke أو descriptor، للحساب تدقيق المجموع. عند تعيين CSUM_V2 أو CSUM_V3 نهاية الكتلة تكون struct jbd2_journal_block_tail			حجم / نوع	إزاحة	رمز تذكري
تدقيق مجموع: معرف قيد الحوادث UUID + كتلة التوصيف، مع تعيين هذا الحقل إلى الصفر.		crc32c(uuid+descr_block)	4	0x00 (00)	t_checksum

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 في jbd2 حين ذلك، تستبدل هذه 4 بايت بأصفار ويتم تعيين علم "escaped" في وسم كتلة التوصيف.

كتلة الإبطال (الإلغاء / النقض...) — Revocation Block

كتلة الإبطال revocation block تستخدم لمنع تكرار الكتلة في إجراء سابق. هذا revoke descriptor يصف سلسلة من الكتل ستلغى من log، بمعنى آخر، يستخدم لتعليم الكتل التي كتبت إلى قيد الحوادث سابقا ولكن لم تعد كذلك. عادة، هذا يحدث إذا كانت كتلة البيانات الوصفية حرة ثم خصصت مرة ثانية ككتلة بيانات ملف؛ في هذه الحالة، تكرار قيد الحوادث 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 إلغاء 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 |................|
1000
```

حجم / نوع		إزاحة	رمز تذكري
journal_header_t	12	0x00 (00)	r_header
__be32	4	0x0C (12)	r_count
__be32 / __be64		0x10 (16)	Blocks[0]

تروبسة كتلة عامة. — عدد بايتات المستخدمة في هذه الكتلة. — كتل للإلغاء Blocks to revoke

بعد حقل r_count 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 4 بايت خلاف ذلك.

حجم / نوع		إزاحة	رمز تذكري	
struct jbd2_journal_revoke_tail ستكون نهاية كتلة الإلغاء، JBD2_FEATURE_INCOMPAT_CSUM_V3 أو JBD2_FEATURE_INCOMPAT_CSUM_V2 بتعيين				
crc32c(uuid+REVOKE_BLOCK)	4	__be32	0x00 (00)	r_checksum

تدقيق مجموع كل من معرف قيد الحوادث UUID +كتلة الإبطال (الإلغاء).

كتلة التنفيذ (أو الإيداع) — Commit Block

كتلة التنفيذ commit block ستكون حارس يشير إلى اكتمال كتابة الإجراء إلى قيد الحوادث.عندما تصل كتلة التنفيذ هذه إلى قيد الحوادث، البيانات المخزنة مع هذا الإجراء 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 00 |.;9........\....|
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]
1000
```

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

1	JBD2_CRC32_CHKSUM	crc32
2	JBD2_MD5_CHKSUM	md5
3	JBD2_SHA1_CHKSUM	sha1
4	JBD2_CRC32C_CHKSUM	crc32c

ملاحظات:

1. ^ أ، ب، ت، ث، مؤشر الفهرسة، أو عقدة الملف (inode كلمة مركبة ومعناها index node)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 وتتعقب الملف أو الدليل، وتتضمن جميع البيانات الوصفية للملف (باستثناء اسم الملف ومضمون الملف) بنية inode يمكن أن تتضمن أيضا بيانات مباشرة inline data في حالة تمكين ميزة Inline Data، وكان مضمون الملف أقل من 60 بايت، أو تتضمن وصلات Symbolic Links، إذا كان حجم الهدف يتناسب مع 60 بايت (راجع الفقرة أعلاه).

عدد inodes الإجمالي مع المساحة المحجوزة لها يتحدد عند إنشاء نظام الملفات أول مرة (حوالى 1% تكرس من أجل inodes)، 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 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 سوف يسمح 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 ببساطة 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)
- 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 المدبات و/أو تخصيص inodes ديناميكية، هذا ينمي نظام الملفات و/أو يرفع عدد inodes. بالمناسبة هناك أنظمة ملفات أصلا لا تستخدم 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 بما فيه الكفاية (مثلا بعدد كتل أكبر من 2^32)، نظام الملفات ext4 سوف يستخدم نمط 64-بت. والأعداد المهمة تصبح بقيم 64-بت.

يمكنك أيضا تحويل نظام الملفات المتواجد على القرص إلى 64 بت: أولا، تحتاج إلى فحص وأمثلة القسم باستخدام e2fsck (بدون وصل) ثم عمل بقية الخطوات (تحويل ثم تمكين دعم تدقيق المجموع):

```
# e2fsck -Df /dev/sda1
Convert the filesystem to 64bit:
# resize2fs -b /dev/sda1
Finally enable checksums support:
# tune2fs -O metadata_csum /dev/sda1
```

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

إزاحة الرقم السحري 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 الاقلاع مثل شفرة VBR

4. ^ إذا لم يكن تعيين الإصدارة الرئيسية إلى الصيغة 2 (الديناميكية) القيم من بايتات 84 فصاعدا قد تكون غير صحيحة. كلمة "الديناميكية" 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 الأنوية 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 البيانات، 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 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 تعدل 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 name_len: قيمة 8 بت لا تحمل إشارة تشير إلى عدد بايتات بيانات المحارف في الاسم.

      ملاحظة: 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 file_type: قيمة 8 بت لا تحمل إشارة للإشارة إلى نوع الملف.

      ملاحظة: 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 name: اسم المدخلة، (غالبا ISO-Latin-1) الذي 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 البانات الوصفية في نظام الملفات وقد يسرع تخزين ريد RAID.
12. ^ عدد الكتل المنطقية المقروءة أو المكتوبة إلى القرص قبل العودة إلى القرص الحالي. يستخدمه محصص 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 stride size, stripe size عمق الشريط, stripe depth طول الشريط, stripe length.

عدد أقراص البيانات في المصفوفة أحيانا يسمى عرض الشريط stripe width، لكن قد تشير أيضا 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 النسبية للقرص: تعني تقنين مساحة القرص أو عدد الملفات الممنوحة 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... (راجع أكثر معلومات الموسوعة الحرة ووثائق نواة لبنكس 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 الملفات 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 Backup superblock.ولأنها احتياطية، نظام الملفات يستخدم 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) الخَرْج 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

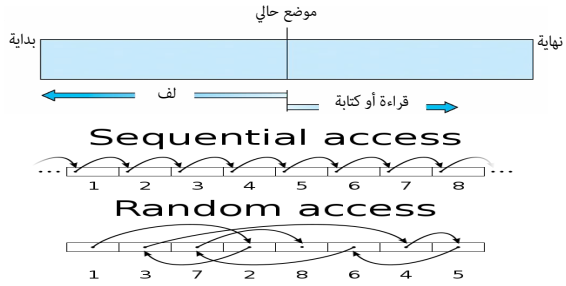
بايتات من 0 إلى 3 تشير إلى الكتلة 2 حيث تقع block bitmap بايتات من 4 إلى 7 تشير إلى الكتلة 3 حيث تقع inode bitmap بايتات من 8 إلى 11 تشير إلى الكتلة 4 حيث يقع inode table.

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 512. ((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 احتياطية من superblock و Group Descriptors، المصفوفة الثنائية للكتل block bitmap ستقع في أول كتلة من المجموعة (جرب dumpe2fs في جهازك)

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

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 الكتلة متطابقة مع الرقم السحري في jbd2 حينذاك 4 بايت تستبدل بأصفار ويتم تعيين علم "escaped" في وسم كتلة التوصيف descriptor block tag

21. ^ أ، ب، ت هذه الحقول كانت سابقا على التوالي h_i_frag ـ h_i_fsize (8 بت) و m_i_frag ـ h_i_fsize (8 بت) أو m_i_frag و m_i_fsize في Masix لكن بما أن (تقنية) fragments ليس مدعوم في لينكس وحنو هيرد و Masix، رقم وحجم fragment. دائما 0. 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 تقبل العنونة، يحددها نظام التشعيل (هذه القيمة أس العدد اثنين، وستكون على الأقل بحجم قطاع على القرص (عادة 512 بايت)، أو حتى بحجم الصفحة الذاكرة). 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 2.6.23. فهارس 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 دقيقة، أو غير كاملة، أو متضاربة. خصوصا في نظام الحاسوب أو قاعدة البيانات. البيانات الملوثة يمكن أن تتضمن أخطاء مثل أخطاء التهجئة، والترقيم، أو بيانات خاطئة مصاحبة للحقل، أو بيانات غير كاملة أو قديمة، أو حتى بيانات مكررة في قاعدة البيانات. 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 الرئيسة RAM المعدلة أثناء كتابة البيانات إلى القرص سوف توسم بالملوثة "dirty" ويجب تخليصها (flushed to) 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 الميزّات feature flags (أيضا feature set...feature toggle, feature switch, feature flipper, conditional feature...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 الميزّات، 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)

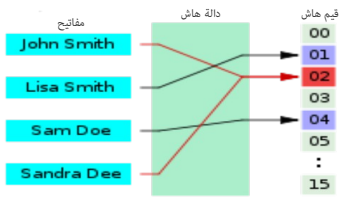
28. ^ أ، ب، تخصيص الكتل المتأخر Delayed Allocation أو التخصيص عند التخليص!ـ Allocate-on-flush، هذه التقنية تعني عند كتابة البيانات إلى القرص بدلا من تخصيصها فورا (على القرص)، سوف تخزن في الخابية cache. والبيانات في الخابية تكتب فقط بعد تخليص الخابية "flush"-ing the cache"، وهذه ستكون فرصة للمحصص الكتل لأمثلة عملية التخصيص باستخدام المديات. هذه الميزة

تستخدمها أنظمة الملفات ext4 ،Btrfs، ZFS، Reiser4، XFS، HFS+ والميزة تشبه كثيرا تقنية قديمة في نظام بيركيلي 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 (flushed to) إلى التخزين نتيجة الضغط على الذاكرة، 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 الفوتوغرافي، 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 الميزة على القرص الأول: hdparm -I /dev/sda | grep -i trim.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 التهيئة، 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 تُحوِّل (أو ترمز) مجموعة كبيرة من البيانات بأي حجم إلى بيانات بحجم أصغر وثابت، تستخدم في جداول هاش، 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 الجدول 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 قد نشأت في الخمسينيات، لا يزال موضوع تصميم دالات هاش قيد البحث. ترتبط دالات هاش بتدقيق المجموع، وتدقيق الأرقام، والبصمات، والدالات العشوائية، والضغط مع فقدان معلومات (ضغط فقود) ورموز تصحيح الخطأ، وشفرة الكتابة ، ودالة هاش الرمزية. وعلى الرغم من 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 وفق للدالة التجزئة، والأفعال المشتقة هي hash out و rehash 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 Bit array, Bit map, Bitmap 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 الذاكرية، 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، 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. 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 تواصل أو تبادل البيانات بين العمليات. ومن بين الوسائل التي يوفرها نظام التشغيل نجد الأنابيب التي تصل خرج عملية بدخل عملية أخرى. هذا جيد إذا كان كلا العمليتين موجود في نفس مجال العملية الأم، التي بدأها نفس المستخدم.لكن، هناك حالات حيث العمليات الموصولة يجب أن تستخدم أنابيب اتصال مسماة أو FIFO 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!ـ 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 بتدفّق 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 المِحرفي 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 المِحرفي 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 text؛ و strings 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. ^ إقفال تنازع الموارد lock contention يحدث حينما تحاول عملية أو خيط حاسوبي الحصول على قفل lock في قبضة عملية أو خيط حاسوبي أخر وكلما كان هناك أقفال متوفرة بحزيئية أدق كان هناك احتمال أقل أن تطلب عملية أو خيط حاسوبي قفل lock في قبضة أخر.(مثال: إقفال صف (تسجيلة) بدلا من جدول كامل، أو إقفال خلية بدلا من صف كامل) .
    - القفل Lock أو استبعاد التشارك mutex آلية تزامن لتأمين تطبيق قيود على الوصول إلى مورد في بيئة حيث يتم تنفيذ عدة خيوط. فيصبح الوصول مؤمن لطالب واحد في نفس الوقت. والقفل صمم لإنفاذ سياسة مراقبة التزامن في الاستثناء المتبادل كما أن الأقفال مستعمل أيضا في أنظمة التشغيل متعددة المستخدمين.
    - استبعاد التشارك أو إقصاء التشارك (Mutual exclusion :mutex) أداة مزامنة تستعملها بعض الخوارزميات المستعملة في البرمجة لتفادي الاستخدام المتزامن للموارد المشتركة، مثل الوصول لمتغير عام، قد تقوم به بعض المقاطع الحرجة. المقطع الحرج هو جزء من البرمجية حيث تسعى عملية أو خيط للوصول فيه إلى مورد مشترك. المقطع الحرج في حد ذاته ليس آلية أو خوارزمية للاستبعاد المتبادل. خوارزميات استبعاد التشارك تسمح بضبط الوصول للبيانات. مثلا، حين يتم تنفيذ روتين معين مرة واحدة في نفس الوقت فلا يقبل التوازي.
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 inode 0 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، أما الكتلة فمفهوم يستخدم على مستوى نظام الملفات، مثلا، في لينكس 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 الدليل.

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 للمتصلة. 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 الدليل الذي يتضمن مدخلات الدليل.

43. ^ أ، ب، مدخلة الدليل المفهرس 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، لكن أنظمة تشغيل أخرى قد تفعل ذلك مستقبلا. 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 (أي غير قابل للقسمة) 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 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

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 : طبقة jbd2 تستخدم ميزة journal_checksum (ربما بشكل غير منتظم) للتأكد من تكامل محتوى قيد الحوادث. وتدعم حاليا تدقيق مجامع MD5، CRC32، أو SHA1، لكن (منذ لينكس 3.0) يبدوا أنها تدعم فقط CRC32. الذي يمكن تغييره بسهولة إلى CRC32c.

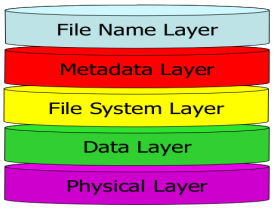
49. ^ عند إنشاء دليل، تعداد الروابط link count سيبدأ من 2 : هذا يعني رابط للدليل الحالي نفسه، وآخر للدليل الأم. (بالإضافة إلى الروابط الفرعية، إن وجدت). وصحيح أن الدليل ملف (خاص)، يتضمن لائحة من أسماء الملفات، لكن (باستثناء المدخلتين "." و "..") وجود روابط صلبة متعددة إلى الدليل، سينشأ عنه حلقات داخل 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 بالكتل المنطقية، وليس عبر 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 (sb.s_log_block_size + 10) ^ 2 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 (بمعني 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. ^ بمكن 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 الهرميّة: مستوى تجريدي أو طبقة تجريدية abstraction layer / abstraction level: طريقة لإخفاء التفاصيل التطبيقية لمجموعة وظيفية محددة (أو ذات تأدية وظيفية محددة).

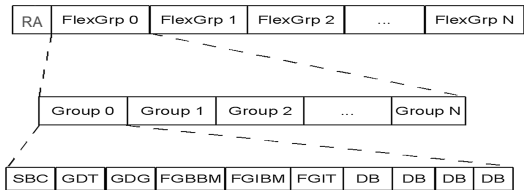


- File Names, Directories
- Inodes
- Superblock
- Sectors, Fragments, Blocks
- VToC, Partitions

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
GDT	Group Descriptor Table	جدول توصيف المجموعات
GDG	Group Descriptor Growth Blocks	الكتل المحجوزة لنمو هياكل Group Descriptor
FGBBM	Flex Group Block Bitmap	المصفوفة الثنائية للكتل في المجموعة المرنة
FGIBM	Flex Group Inode Bitmap	المصفوفة الثنائية للمؤشرات الفهرسة في المجموعة المرنة
FGIT	Flex Group Inode Table	جدول مؤشرات الفهرسة في المجموعة 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 : **dumpe2fs /dev/sda1 > dumpfile**

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 Block Bitmap عند الكتلة 956، و 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
```

إيجاد الفهارس مباشرة بدون استخدام dumpe2fs. كتل Group Descriptors تبدأ من الكتلة الثانية، وأول ثلاثة قيم 32-بت ستكون مواقع 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. وهذه 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 ؟ ـ ((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. 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. (لتمكين sparse_super)
- تتضمن Data Block Bitmaps، و inode Bitmaps، و inode Tables لجميع مجموعات الكتل الأخرى (16-31) في مجموعة الكتل المرنة هذه.

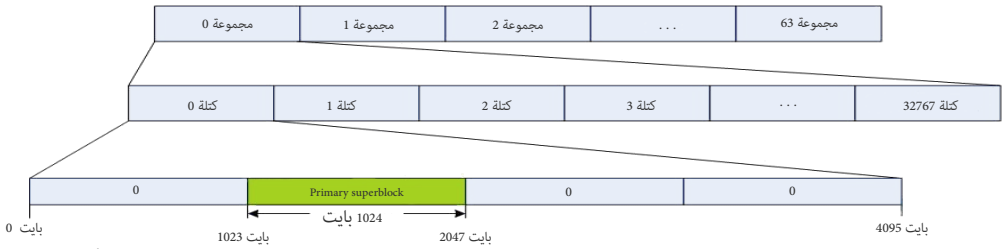
```
Group 16: (Blocks 524288-557055) [INODE_UNINIT, ITABLE_ZEROED]
  Checksum 0x8ab4, unused inodes 8144
  Block bitmap at 524288 (+0), Inode bitmap at 524304 (+16)
  Inode table at 524320-524828 (+32)
  24592 free blocks, 8144 free inodes, 0 directories, 8144 unused inodes
  Free blocks: 532464-557055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

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%، ويمكن تغييرها بخيار 0 m- مع أداة tune2fs. (لكن لا ينصح بذلك في الأنظمة الصغرى) راجع استخدام **MKE2FS / TUNE2FS**

> -m reserved-blocks-percentage
> Specify the percentage of the filesystem blocks reserved for the super-user. This avoids fragmentation, and allows root-owned daemons, such as syslogd(8), to continue to function correctly after non-privileged processes are prevented from writing to the filesystem. The default percentage is 5%.

59. ^ 32 بت تشير إلى العدد الإجمالي للكتل الحرة، والتي تشمل عدد الكتل المحجوزة (s_r_blocks_count). هذا سيكون حاصل جميع الكتل 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 بنية superblock. لاحظ أن هذه القيمة دائما 0 في أنظمة الملفات التي تستخدم حجم كتلة أكبر من 1 كيلوبايت، ودائما 1 في أنظمة الملفات التي تستخدم حجم كتلة 1 كيلوبايت.superblock دائما تقع عند إزاحة البايت 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 أنظمة الملفات 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 1 كيلوبايت، مجموعة الكتل 0 تبدأ في الكتلة 1، وعند استخدام حجم الكتلة الأكبر، (أنظر للخطاطة أسفل) تبدأ مع الكتلة 0. (راجع حقل s_first_data_block في superblock).



مثال : 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، توسم الكتلة 0 بالمستخدمة ولا تستخدم. والكتلة 1 تتضمن Primary superblock، المجموعة 0 تتضمن أيضا كتل Group descriptors و Reserved GDT blocks و Block bitmap، و Inode bitmap، و Inode table، و 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 Group descriptors و Reserved GDT blocks و Block bitmap، و Inode bitmap، و Inode table، و 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 توصيف المجموعة الخاص Group descriptor، ولذلك الترتيب قد يختلف عن هذا.
- في حال تمكين sparse_super + flex_bg، أغلب مجموعات الكتل سوف تتضمن فقط كتل البيانات.

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 (EXT2_GOOD_OLD_FIRST_INO).لكن 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 (EXT2_GOOD_OLD_INODE_SIZE). 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 للنظام) يمكن استخدام هذا في إعادة بناء نظام الملفات من أية نسخة superblock.
70. ^ 32 بت (قناع بت) أعلام الميزّات المتوافقة. تطبيق نظام الملفات سيكون حر في دعم هذه الميزّات أو لا، 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. تطبيق 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، تطبيق 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 نظام الملفات 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
      - لكن إذا تم تعيين علم EA_INODE، هذا inode يخزن قيمة الخاصية الممتدة والحقل (i_ctime) يحوي 32 بت المنخفضة من التعداد المرجعي للقيمة الخاصية.
    - 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 تميكن 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)
    - 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_lo + (i_blocks_hi << 32) على القرص.
    - في حال تعيين huge_file مع تعيين EXT4_HUGE_FILE_FL IS، هذا الملف يستهلك كتل نظام الملفات (i_blocks_lo + i_blocks_hi << 32) 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) مؤلّف 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. ^ الأحراء هو عملية ذرية ؛ رسالة، تعديل للبيانات، أو إجراء أخر يضمن إما تنفيذه كاملا، أو لا يتم تنفيذه إطلاقا (مثل، إحراء 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 الكتلة العليا 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 هذا 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 (وهو عدد الكتل في المجموعة)، نحصل على أرقام الكتل التالية (التي تظهر في خرج mkfs عند إنشاء نظام الملفات)

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

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! للملفات

87. ^ حقول تخزين تدقيق المجامع الجارية *h_chksum في ترويسة الكتلة التنفيذ Commit Block :

في حال تعيين FEATURE_COMPAT_CHECKSUM ـ(checksum v1)، حقول h_chksum تستخدم في تخزين تدقيق مجموع كتل descriptor و data .

في حال تعيين FEATURE_INCOMPAT_CSUM_V2 ـ(checksum v2)، حقول h_chksum تستخدم في تخزين crc32c(uuid+commit_block). كل كتلة بيانات وصفية في قيد الحوادث تحصل على تدقيق مجموع خاص، وتدقيق مجامع data يخزن في journal_block_tag (في descriptor). حقول h_chksum الأخرى لا تستخدم.

في حال تعيين FEATURE_INCOMPAT_CSUM_V3، كتلة 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 ميزّات متنافية (أي، لا يمكن أن تقع في نفس الوقت) [هذه المعلومات من ملف: kernel-jbd.h]

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

89. ^ حقل h_chksum[JBD2_CHECKSUM_BYTES] يشير إلى مساحة من 32 بايت من أجل تخزين تدقيق المجامع. في حال تعيين JBD2_FEATURE_INCOMPAT_CSUM_V2 أو تعيين JBD2_FEATURE_INCOMPAT_CSUM_V3، أول __be32 ستكون تدقيق مجموع بحساب معرف journal UUID وكامل كتلة التنفيذ commit block، وهذا الحقل سيكون صفر. وفي حال تعيين JBD2_FEATURE_COMPAT_CHECKSUM، أول __be32 ستكون crc32 لجميع الكتل التي تم كتبتها بالفعل إلى الإحراء.

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

91. ^ 16 بت تشير إلى تَعْداد inodes الغير مستخدمة، في حال التعيين، لا نحتاج 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؛ (sb_uuid+group+desc) crc16 في حال تعيين ميزة RO_COMPAT_GDT_CSUM، أو (sb_uuid+group_desc) crc32c و 0xFFFF في حال تعيين ميزة RO_COMPAT_METADATA_CSUM

93. ^ في النسخة v2 .كل كتلة بيانات وصفية في قيد الحوادث تحصل على تدقيق مجموع خاص. و block tags في جدول التوصيف تتضمن تدقيق مجامع لكل كتلة بيانات في قيد الحوادث.

94. ^ النسخة v3 مثل V2، لكن حجم journal block tag ثابت بغض النظر عن حجم أعداد الكتل.

95. ^ أ، ب، ت، ث، ج، ح، ميزة meta_bg تسمح بإعادة تحجيم نظام الملفات في وضع المتصل resized on-line دون الحاجة إلى حجز مساحة للنمو توصيف الكتل 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، و ألأنويه الأحدث يمكنها آليا تعيين هذه الميزة عند الحاجة عند عمل إعادة تحجيم في وضع المتصل online resize ولا تتوفر مساحة محجوزة في resize inode

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،

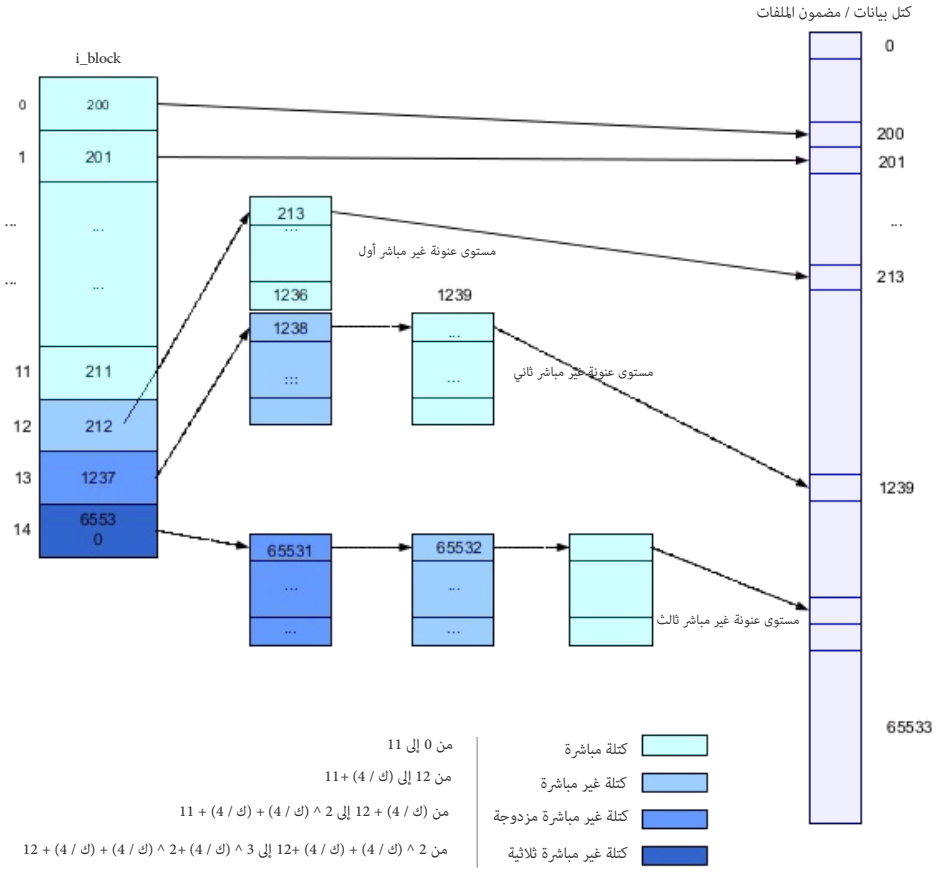
97. ^ أ، ب، ت، عنونة الكتل المباشرة والغير مباشرة تملك 12 رابط مباشر، ورابط غير مباشر، ورابط غير مباشر مزدوج، ورابط غير مباشر ثلاثي (المجموع 15، لأن 60 بايت ÷ 4 = 15) هنا كلمة غير مباشر(ة) Indirect تعني أن الكتلة في العنوان ليست كتلة بيانات بل كتلة أخرى (مستوى آخر) تعرض عناوين كتل أخرى، وقد تملك روابط 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 مدبات extents.

حقل i_block هو مصفوفة تتضمن عناوين للكتل البيانات المرتبطة بمؤشر الفهرسة. 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 من 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 من: 12-1035
- 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)

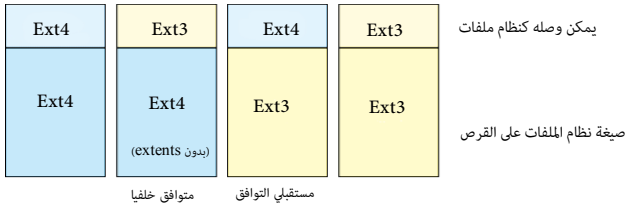


مخطط ربط الكتل (15 × "32-بت") أو العنونة الغير مباشرة كان يستخدم في ext2/3 والآن هذا المخطط يستخدم فقط للتوافق خلفيا ؛ عند التحويل بين ext3 و ext4

- 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 1 كيلوبايت، مع حجم الكتلة 4 كيلوبايت، 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.

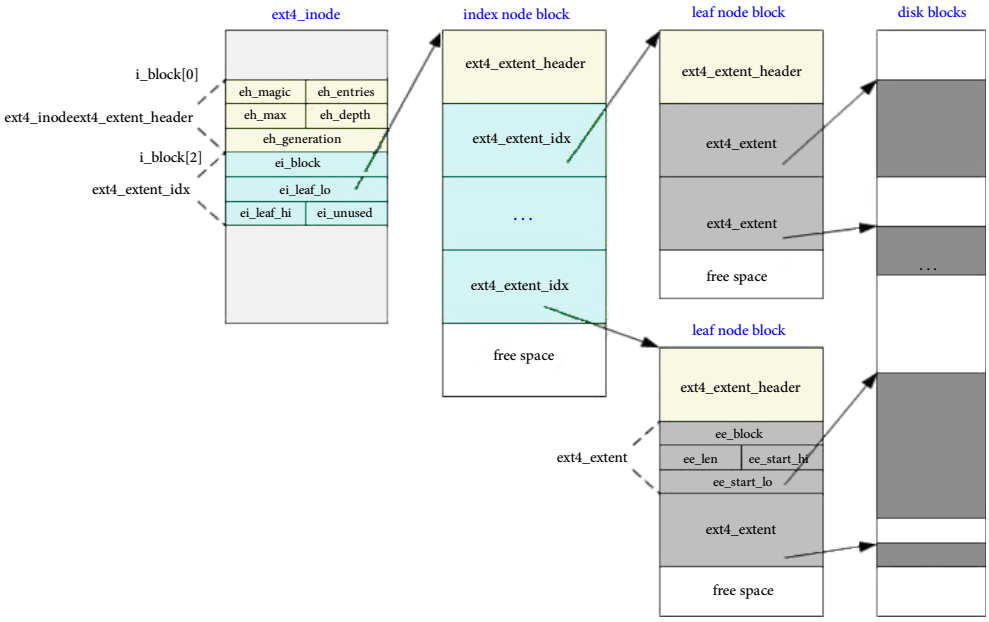
98. ^ أ، ب، ت، ث، ج، بطانة، حشو، حشوة padding تعني محارف إضافية مثل الفراغات تضاف إلى نهاية تسجيلة حتى تصبح بطول ثابت. عادة هذا يستخدم في محاذاة هياكل البيانات وفي علم التشفير.

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 : كتلة : إزاحة)
- مدخلة واحدة لكل رتل / تنفيذ متتالي consecutive run
- المديات extents الأربعة الأولى ستكون داخل inode
- هذه بنية شجرية متراصة (أو مدمجة) تحمل بسرعة
- 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 مستويات: رقم الكتلة المنطقية يمكن أن يكون على الأكثر 2^32، وأصغر رقم n يمكن أن يفي بشرط التالي، سيكون خمسة 5 :

blocksize - 12)/12)^n) >= 2^32)))*4

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 32768 – ee_len. 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 كتلة واحدة في مجموعة الكتل، إذن، الكتل التي يمكن تعقبها باستخدام كتلة block bitmap ستكون (8x(block size)) أو 32 كيلوبايت في أنظمة ملفات حجم الكتلة 4 كيلوبايت. block bitmap، إلى جانب الكتل المحجوزة من أجل inodes و inode bitmap (نظام EXT عادة يخصص inode واحد لكل 4 كتل في المجموعة) بالإضافة إلى نسخ superblock والبيانات الوصفية للنظام الملفات الأخرى التي عادة تخزن مباشرة قبل كتل البيانات في مجموعة الكتل. جميع هذه البيانات الوصفية تعني أنك لن تجد أكثر من 32 كيلوبايت من كتل البيانات المتماسة في نظام ملفات EXT وذلك فقط إذا كانت مجموعة الكتل المعنية حاليا شاغرة.

في سياق حقل حجم المديات في ext4. الحقل بقيمة 16-بت، لكن بت العليا محجوز، هذا يعني أن المدى يمكن أن يتضمن فقط 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 قيمة حجم المدى محجوز ؟ الجواب كي يستطيع نظام الملفات وسم بعض المديات بالغير مهيئة لكنها محجوزة "uninitialized but reserved".استراتيجية التخصيص المسبق هذه "preallocation" تسمح للنظام ext4 منع الملفات الأخرى من استخدام كتل معينة، إذا ظن النظام أن الملف سيحتاج هذه الكتل مستقبلا، ومن ثم تجنب تجزئة الملف الذي سينمو.

لذلك 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 ؟. للإجابة على ذلك، مطوري ext4 أضافوا حالة خاصة تقول أن القيمة 0x8000 تعني مدى مخصص allocated extent لكامل كتل 32 كيلوبايت في مجموعة الكتل. جميع القيم الأخرى في حالة تعيين بت العليا high bit تعني مدى مخصص مسبقا preallocated لكن بدون تهيئة uninitialized وطول المدى تحدده 15 بت الأخرى في حجم المدى. لكن ذلك يعني إمكانية فقط تخصيص مسبقا ما يصل إلى 1 – 2^15 كتلة، أو 32767، أي أقل بكتلة واحدة من العدد الأقصى للكتل في مجموعة الكتل. وهذا بالضبط (ما يحاول اخبرانا به) أو ما جاء في نص الشفرة والتعليق أعلاه، إذن الجواب باختصار، قيمة حجم المدى 0x8000 تعني مدى مخصص allocated extent بطول كتل من 32 كيلوبايت. 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.

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

أدوات	العديد من هذه الأدوات ترتكز على مكتبة libext2fs library / تنبيه : الاستعمال الخاطئ سوف يتلف نظام الملفات !	
badblocks	search a device for bad blocks	البحث عن الكتل المعيبة (التالفة) على القرص
blkid	locate/print block device attributes	تحديد مكان / طباعة خصائص جهاز الكتل (مثل dev/sda1)
chattr	change file attributes on a Linux file system	تغيير خصائص الملفات على نظام ملفات لينكس
debugfs	used to manually view or modify internal structures of the file system	عرض أو تعديل هياكل نظام الملفات الداخلية
dumpe2fs	which prints superblock and block group information.	طباعة معلومات كتلة العليا للنظام الملفات ومجموعة الكتل
E2freefrag	report free space fragmentation information	تقدم تقريرا عن معلومات تجزئة المساحة الحرة
e2fsck	an fsck program that checks for and corrects inconsistencies	إحدى أدوات fsck التي تتفحص وتصحح التضاربات
E2image	save critical ext2/ext3/ext4 filesystem metadata to a file	يحفظ في ملف البيانات الوصفية الحرجة لأنظمة ملفات ext2/ext3/ext4
e2label	change the label on an ext2/ext3/ext4 filesystem	يغير لصيقة (اسم / عنوان) على أنظمة ملفات ext2/ext3/ext4
e2undo	replay an undo log for an ext2/ext3/ext4 filesystem	إعادة تشغيل سجل التراجع undo log لأنظمة ملفات ext2/ext3/ext4
e4defrag	online defragmenter for ext4 filesystem	برنامج لإلغاء التجزئة في وضع المتصل من أجل نظام ملفات ext4
filefrag	report on file fragmentation	تقدم تقريرا عن تجزئة الملفات
findfs	find a filesystem by label or UUID	إيجاد نظام الملفات عن طريق لصيقة أو المعرف label / UUID
Findsuper	quick hacked up program to find ext2 superblocks	برنامج لإيجاد الكتل العليا ext2
logsave	save the output of a command in a logfile	يحفظ خرج الأمر إلى ملف السجل
lsattr	list file attributes on a Linux second extended file system	سرد خصائص الملفات على نظام ملفات لينكس ext2
Mke2fs	used for creating ext2, ext3 and ext4 file systems	يستخدم في إنشاء أنظمة ملفات ext2/ext3/ext4
Resize2fs	which can expand and shrink ext2, ext3 and ext4 file systems	توسيع وتقليص (إعادة تحجيم) أنظمة ملفات ext2/ext3/ext4
tune2fs	used to modify file system parameters	تعديل معاملات نظام الملفات

104. ^ أ، ب، ميزّات نظام ملفات ext4 :

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 بدون خطر إتلاف البيانات الوصفية.

مجموعة أعلام الميزّات الغير متوافقة Incompatible feature set: نظام التشغيل لا يجب أن يصل (نظام الملفات) إذا كان لا يدعم هذه الميزّات. وتطبيق نظام الملفات الذي لا يدعم هذه الميزات سيكون غير قادر على استخدام نظام الملفات بالشكل الصحيح. مثال، إذا كان ضغط البيانات مستخدم وملف تنفيذي صار غير صالح للاستعمال بعد قراته من القرص لأن النظام لا يعرف كيفية فك ضغطه.

مجموعة أعلام الميزّات المتوافقة – في وضعية القراءة فقط Read-only compatible feature set: ينبغي وصل (نظام الملفات) في وضعية القراءة فقط إذا كانت الميزّات بدون دعم.

مجموعة الميزّات التجريبية (الاختبارية)! 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	*	*		s_feature_compat	EXT2_FEATURE_COMPAT_DIR_PREALLOC	0x1	dir_prealloc
"imagic inodes"	*	*			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 !	*	*			EXT2_FEATURE_COMPAT_LAZY_BG	0x40	lazy_bg
"exclude inode"					EXT4_FEATURE_ COMPAT_EXCLUDE_INODE	0x80	exclude inode
"Exclude bitmap"	*	*			EXT2_FEATURE_COMPAT_EXCLUDE_BITMAP	0x100	snapshot_bitmap
تمكين ميزة النسخ sparse_super2		*	ext4, 3.16		EXT4_FEATURE_COMPAT_**SPARSE**_SUPER2	0x200	sparse_super2
النظام يستخدم ضغط البيانات	*	*		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 !	*	*			EXT3_FEATURE_INCOMPAT_RECOVER	0x4	needs_recovery
النظام يملك جهاز قيد حوادث منفصل	*	*			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_FEATURE_INCOMPAT_DIRDATA	0x1000	dirdata
بذرة تدقيق مجموع البياتات الوصفية في SB		*	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
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
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					EXT4_FEATURE_RO_COMPAT_BTREE_DIR	0x4	
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					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		*			EXT4_FEATURE_RO_COMPAT_REPLICA	0x800	replica
صورة نظام ملفات للقراءة فقط		*			EXT4_FEATURE_RO_COMPAT_READONLY	0x1000	read-only
نوع جديد من أجل تعقب حصص القرص		*	ext4, 4.5		EXT4_FEATURE_RO_COMPAT_PROJECT	0x2000	project

تنبيه : بعض الميزّات (حتى وإن كانت قديمة!) ربما ما زالت في مرحلة الاختبار ولا ينصح باستخدامها إلا على الأجهزة الاختبارية. (جدول خاص)

105. ^ ولأن lost+found ينشأ آليا، إذن أول عنوان inode في مساحة المستخدم (للاستعمال الشخصي) سيكون 12 مثال:

```
# 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 inode للدليل 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 (مثل 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 البايت في الكتلة المحددة). بالنسبة لأنظمة لينكس الحالية، هناك كتلة واحدة فقط، ولا يتم تعيين حقل الكتلة. عدة خصائص يمكنها تشارك نفس القيمة. بالنسبة للخاصية inode attribute 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 hash value الخاصة 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 تكون 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)

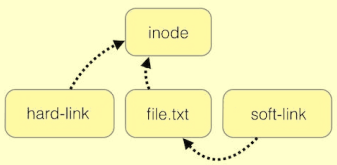
الوصلة الصلبة أو الرابط الصلب Hard Link عبارة عن مدخلة دليل تربط بين الاسم والملف (تحديدا inode number) على نظام الملفات. جميع أنظمة الملفات التي ترتكز على الأدلة يجب أن تملك (على الأقل) رابط صلب واحد يمنح إلى الاسم الأصلي لكل ملف. تعبير "hard link" عادة يستخدم فقط في أنظمة الملفات التي تسمح بأكثر من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

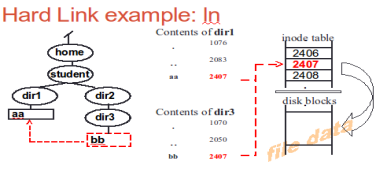
عملية إنشاء رابط صلب تعني منح ملف واحد أسماء متعددة (مثلا، أسماء مختلفة في أدلة مختلفة) وجميعها مستقلة لكنها مرتبطة بنفس البيانات على القرص. ولا أحد منها يعتمد على الأخر, وهذا له تأثير الاسم المستعار أو الكنية؛ مثلا، إذا فتحنا الملف عبر أحد أسماءه، وتم تعديل المحتوى، التغييرات ستظهر عند فتح الملف من 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 بمنح منفذ مباشر إلى اسم ملف أخر ضمن دليل معين.

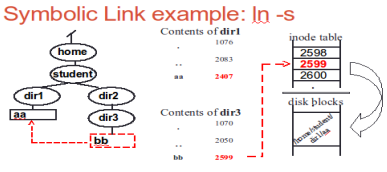
الاسم المضمن أو المشار إليه بواسطة الوصلة الرمزية يمكن أن يكون رابط صلب أو وصلة رمزية أخرى. هذا أيضا له تأثير aliasing، لكن بطريقة مختلفة.

ولأن inode بنية بيانات 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 Symbolic Link تعني أي ملف يتضمن مرجع reference إلى ملف أو دليل أخر في شكل مسار نسبي أو مطلق الذي يأثر على ترجمة أو تحليل اسم المسار pathname resolution.

الوصلة الصلبة (مدخلة دليل)	الوصلة الرمزية (ملف خاص)
◦ كل inode يمثل ملف واحد يمكن أن يقترن بعدة أسماء (لا يخضع لاسم الملف).	◦ الوصلة الرمزية هي ملف خاص يتضمن اسم ملف (يشير إلى ملف أخر) "مضمون" الملف هو هدف الوصلة
◦ الروابط الصلبة يمكنها فقط الإشارة إلى الملفات. هذا يمنع الدورات cycles في شجرة الدليل (باستثناء "." و"..").	◦ الوصلة الرمزية يمكنها الإشارة إلى أي نوع من الملفات (بما في ذلك، الدليل)
◦ الروابط الصلبة تملك نفس أرقام مؤشر فهرسة inode numbers	◦ الوصلة الرمزية تملك أرقام مؤشر فهرسة مختلفة inode number
◦ الروابط الصلبة تستخدم فقط داخل نظام الملفات الواحد.	◦ الوصلة الرمزية يمكنها الإشارة إلى الملفات على نظم الملفات الأخرى
◦ الروابط الصلبة لا تدعمها جميع أنظمة الملفات.	◦ 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 مؤشرات 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.
◦ تعداد الروابط i_links_count يزداد مع كل إنشاء رابط ويتناقص مع كل حذف للملف (رابط) من الدليل. inode والبيانات المرتبطة تحذف فقط عندما يصبح تعداد المراجع 0.	◦ 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 مدلىً أو يتم "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





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/2080d  Inode: 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، وفي الوصلة الرمزية الحجم 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]
```

```
# 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) :

```
# sudo 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)
```

112. ^ أ، ب، ت، طريقة إصلاح الكتلة المتضررة super block عن طريق البديلة (أي الاحتياطية) Alternative Superblocks (نظام ملفات ext4)

1. الإقلاع عبر قرص التنصيب Live CD/USB ثم تحديد قسم لينكس على القرص HDD / SSD (مثال: dev/sda1)
2. تأكد من المشكلة (تنبيه : بدون وصل نظام الملفات أي القسم)
3. تحديد موقع النسخ البديلة / الاحتياطية
4. 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 superblock يشير إلى alternative superblock،

إذا فشلت الخطوة 4، حاول تكرارها مع القيمة في الخطوة 3

1. `# sudo fdisk -l`
2. `# sudo fsck.ext4 -v /dev/sda1`
3. `# sudo mke2fs -n /dev/sda1`

   ```
   Superblock backups stored on blocks:
   32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208
   ```
4. `# sudo e2fsck -f -b 98304 /dev/sda1`

1. أيضا يمكن وصل نظام الملفات باستخدام الكتلة العليا البديلة، بإنشاء نقطة الوصل المسماة ثم وصل نظام الملفات عن طريق الكتلة 98304
2. فحص نظام الملفات والكتل المعيبة bad blocks (باستخدام الطرفية في القرص LiveCD):

1. `# mkdir -p /mnt/linux1`

   ```
   # mount -o sb=98304 /dev/sda1 /mnt/linux1
   ```
2. `# sudo umount /dev/sda1`

   ```
   # sudo fsck.ext4 -fck /dev/sda1
   ```

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

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

استخراج كتل journal inode :

```
# sudo 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 (6324224 + 13972) = 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
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
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 تعدد 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 تحريرها، 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 قيد الحوادث journal، خصوصا في حالة فشل العتاد والحل دئما 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) لكن أنظمة الملفات المتعددة، لا يمكن أن تتشارك جهاز قيد الحوادث فكرة نظام الملفات المزود بقيد حوادث، تقوم أساسا على تسجيل التغييرات في نظام ملفات (بكتابة البيانات مسبقا write-ahead) في قيد (أو سِجِلّ) هذا الأخير عادةً يكون "سجل دوري" محجوز في منطقة محددة، قبل كتابة البيانات فعليًا إلى نظام الملفات. بهذه الطريقة، إنشاء وحذف وتعديل الملف يصبح إجراء. وتطبق هنا نفس فكرة ذرية الإجراءات الموجودة في نظم إدارة قواعد البيانات DBMS. 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 البياتات 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 أخر 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).

خيارات إنشاء جهاز قيد حوادث خارجي :

0. إنشاء جهاز قيد حوادث خارجي external journal device ثم ربط نظام الملفات بالجهاز
1. أو إنشاء نظام الملفات واختيار جهاز قيد الحوادث الخارجي في نفس الوقت.
2. أو تغيير قيد الحوادث الداخلي إلى خارجي، في نظام ملفات موجود فعليا على القرص، تحتاج أولا إلى فصل unmount نظام الملفات ثم تنفيذ بقية الأوامر

0. `# mke2fs -O journal_dev /dev/block_device_name`

   `# mke2fs -t ext4 -J device=/dev/journal_device_name`

1. `# mke2fs -t ext4 -J device=/dev/journal_device_name  /dev/block_device_for_new_fs`

2. `# umount /dev/existing_fs`

   `# tune2fs -O ^has_journal /dev/blk_dev_for_existing_fs`

   `# tune2fs -o journal_dev -j -J device=/dev/device_name /dev/blk_dev_for_existing_fs`

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 المجموعة 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.

741 = 388609 - 389350

```
# 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|
[...]
```

بايتات من 4 إلى 7 . (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 (62218 = 0xF30A) هذا الرقم للتمييز بين مختلف تطبيقات المديات. بسبب إضافة ميزّات جديدة. الرقم السحري قد يتغير للضمان التوافق الخلفي .

بايتات من 42 إلى 43 تشير إلى عدد المديات extents، و(1 = 0x0001) تعني امتلاك الملف مدى واحد فقط.

بايتات من 44 إلى 45 تشير إلى العدد الأقصى للمديات، و(4 = 0x0004)، يعني أن العدد الأقصى 4 مديات داخل inode

بايتات من 46 إلى 47 تشير إلى عمق الشجرة (0 = 0x0000).

بايتات من 48 إلى 51 تشير إلى هوية أو رقم توليد الشجرة Generation ID. (0 = 0x00000000)

سوف نناقش "Depth of tree" و "Generation ID" في المقالات القادمة.

بايتات من 52 إلى 55 تشير إلى رقم الكتلة المنطقية Logical block number (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 كان 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 address (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 0x0000**00188305** (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 |................|
1000
```

بما أنه لا توجد مديات إضافية (بالنسبة للترويسة المديات السابقة)، 36 بايت التالية في inode ستكون شاغرة null.. (ملاحظة: الحقول الغير مستخدمة يمكن أن تستخدم في إخفاء البيانات).

لاحظ ماذا يحدث داخل inode وكتلة / كتل البيانات عند حذف الملف المعني 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 |................|
1000
```

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  |................|
0080  1c 00 00 00 8c db 9f 58  8c db 9f 58 90 6f c3 55  |.......X...X.o.U|
0090  a6 e0 d7 5b 90 6f c3 55  00 00 00 00 00 00 00 00  |...[.o.U........|
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  |................|
0000  a4 81 00 00 e8 05 00 00  49 13 cc 5b 04 f5 8a 58  |........I..[...X|
0100
```

مضمون inode بعد حذف الملف :

```
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  |................|
0080  1c 00 00 00 c4 b1 5f 13  c4 b1 5f 13 08 ef 69 e0  |......_..._...i.|
0090  a6 e0 d7 5b 90 6f c3 55  00 00 00 00 00 00 00 00  |...[.o.U........|
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  |................|
0100
```

ماذا حدث في inode عند إلغاء تخصيصها deallocated. ؟

أولا، لاحظ بعد الحذف، إلى جانب تغير قيم التاريخ الأخرى ظهور تاريخ 0x5BD939A8 الذي يترجم إلى 1540962728@ -d date 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 (مثال 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 للقراءة read، للكتابة write، أو للتنفيذ 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 خصائص خاصة properties.

بتات الأخيرة، من 12 إلى 15 من أجل التعريف 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
```

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 6d09 2a3f f607 2a3f .....@..m.*?..*?
0016: 8107 2a3f 0000 0000 f401 0100 404e 0000 ..*?........@N..
0032: 0000 0000 0000 0000 2e38 0000 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).

(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 إلى الملف 0x3f2a096d، الذي يترجم إلى August 1, 2003 at 06:32:13 UTC. (عن طريق: 1059719533@ date -d $) 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 علام 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). بايتات من 92 إلى 95 تشير إلى مؤشر الكتلة الغير مباشر(ة) المزدوج double indirect block pointer في الكتلة 15,417 (0x00003c39)

مضمون كلتا هذه الكتل 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
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 inode تخزن في المصفوفة الثنائية inode bitmap التي في نفس المجموعة وينتمي إليها inode. توصيف المجموعة group descriptor يتضمن عنوان كتلة المصفوفة الثنائية inode bitmap، التي في المثال كانت الكتلة 3، وكان من مضمونها التالي:

```
# blkcat -f linux-ext3 ext3.dd 3 | xxd
000: ff[f7] fcff 1f00 0000 00c8 0000 0000 0000 ................
```

من أجل تحديد البايت 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 inode 16 في البايت f7). بت inode 16 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
```

مثال آخر لكن في ext4

```
$ dd if=ext4.dd bs=256 skip=7105 count=1 | hexdump -Cv
00000000  b4 81 e8 03 6e 2a 72 00  84 cb c6 5b 5d 35 5b 00  |....n*r....[]5[.|
00000010  39 3d 35 5b 00 00 00 00  e8 ee 01 00 18 39 00 00  |9=5[.........9..|
00000020  00 00 08 00 01 00 00 00  0a f3 03 00 04 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 02 00 00 00  |................|
00000040  00 02 00 00 00 02 00 00  00 00 48 12 00 00 04 00  |..........H.....|
00000050  23 03 00 00 00 e0 12 00  00 00 00 00 00 00 00 00  |#...............|
00000060  00 00 00 00 80 54 44 4d  00 00 00 00 00 00 00 00  |.....TDM........|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  20 00 00 00 34 25 12 bc  88 ee 48 29 f0 c7 10 3f  | ...4%....H)...?|
00000090  14 3d 35 5b 9c 17 fa 81  00 00 00 00 00 00 00 00  |.=5[............|
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000100
```

```
$ sudo debugfs -R "stat <549538>" /dev/sda1
Inode: 549538   Type: regular    Mode:  0664   Flags: 0x80000
Generation: 1296323712    Version: 0x00000000:00000001
User:  1000   Group:  1000   Size: 7481966
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 14616
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x5b35955d:bc122534 -- Fri Jun 29 03:11:41 2018
 atime: 0x5bc6cb84:3f10c7f0 -- Wed Oct 17 06:41:24 2018
 mtime: 0x5b353d39:2948ee88 -- Thu Jun 28 20:55:37 2018
crtime: 0x5b353d14:81fa179c -- Thu Jun 28 20:55:00 2018
Size of extra inode fields: 32
EXTENTS:
(0-511):1183744-1184255, (512-1023):1198080-1198591, (1024-1826):1236992-1237794
(END)
```

```
$ sudo istat /dev/sda1 549538

inode: 549538
Allocated
Group: 67
Generation Id: 1296323712
uid / gid: 1000 / 1000
mode: rrw-rw-r--
Flags:
size: 7481966
num of links: 1

Inode Times:
Accessed: Wed Oct 17 06:41:24 2018
File Modified:      Thu Jun 28 20:55:37 2018
Inode Modified:     Fri Jun 29 03:11:41 2018

Direct Blocks:
258826 0 0 1183744 1198080 1236992 0 0
```

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: "directory" , "simple" , "a" , "is" , "this"

```
$ 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 مع تمرير خرج إلى hexdump تظهر مدخلات الدليل بترتيبها الفعلي:

```
debugfs -R "cat <518977>" /dev/sda1 | hexdump -C
00000000  41 eb 07 00 0c 00 01 02  2e 00 00 00 08 ee 05 00  |A...............|
00000010  0c 00 02 02 2e 2e 00 00  42 eb 07 00 0c 00 04 01  |........B.......|
00000020  74 68 69 73 cc ee 07 00  0c 00 02 01 69 73 00 00  |this........is..|
00000030  cd ee 07 00 0c 00 01 01  61 00 00 00 33 ef 07 00  |........a...3...|
00000040  10 00 06 01 73 69 6d 70  6c 65 00 00 56 ef 07 00  |....simple..V...|
00000050  b4 0f 09 01 64 69 72 65  63 74 6f 72 79 00 00 00  |....directory...|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001000
```

على اليمين، تظهر أسماء الملفات (بشفرة 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 ".."، نوع الملف 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.

راقب ماذا يحدث بعد حذف الملف "simple" من الدليل:

```
debugfs -R "cat <518977>" /dev/sda1 | hexdump -C
     00 01 0 203 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 1c 00 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 |................|
1000
```

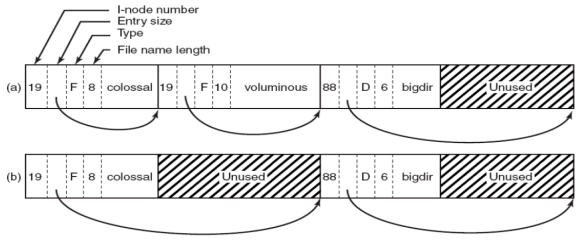
طول مدخلة الملف "a"، الذي كان سابقا 12 بايت، أصبح الآن 28 بايت (0x1C)، بعد استهلاك مدخلة الملف المحذوف "simple". لكن طول اسم الملف في مدخلة الملف "a" لا يزال 1 بايت فقط. بقية المدخلة مجرد مساحة مهملة "slack"، لكنها تحتفظ بالمدخلة القديمة للملف المحذوف. وهذا هو السلوك المعياري عند حذف الملفات في أنظمة 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 can be carved for.

مثال آخر، مع (ext2)

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

ترجمة الحقول :

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



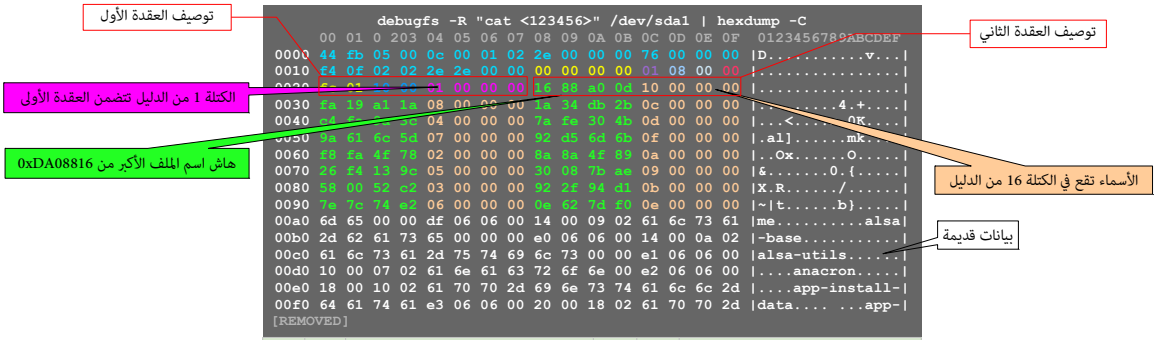
(a) A Linux directory with three files
(b) After the file voluminous has been removed

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:

```
debugfs -R "cat <518977>" /dev/sda1 | hexdump -C
     00 01 0 203 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 00 01 01 61 00 00 00 33 ef 07 00 |........a...3...|
0040 10 00 03 01 6e 65 77 70 6c 65 00 00 56 ef 07 00 |....newple..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 |................|
1000
```

طول مدخلة ملف "a" عاد مرة أخرى 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 فراغات المساحة 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 :




```
debugfs -R "cat <123456>" /dev/sda1 | hexdump -C
     00 01 0 203 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 fa 01 76 ee e1 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 e4 ... ... 04 00 00 00 7a fe 30 4b 0d 00 00 00 |...<....zK....|
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 10 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]
```

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 24. 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
1	"zero hash" 0x00000000
16	0xDA08816
8	0x1AA119FA

إزاحة الكتلة	قيمة هاش
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 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 30، وتملك قيمة الهاش 0x1AA119FA.

يمكن طرح هذه المعلومات باستخدام debugfs مع خيار htree_dump

```
debugfs -R "htree_dump /usr/share/doc" /dev/sda1
Root node dump:
            Reserved zero: 0
            Hash Version: 1
            Info length: 8
            Indirect levels: 0
            Flags: 0
Number of entries (count): 16
Number of entries (limit): 508
Entry #0: Hash 0x00000000, block 1
Entry #1: Hash 0x0da08816, block 16
[REMOVED] . . .
```

كل مدخلة تتضمن قيمة الهاش الأصغر مع كتلة دليل خاصة بالعقدة node

توصيف العقدة الأول first node لا يحتاج إلى حد أدنى ويجب أن يكون 0

```
Entry #15: Hash 0xf07d620e, block 14

Entry #0: Hash 0x00000000, block 1
Reading directory block 1, phys 1582183
394970 0x082a9e46-523d4ecf (12) acl
394976 0x0d65a9d2-bd35910e (20) alsa-utils
[REMOVED] . . .
Entry #1: Hash 0x0da08816, block 16
Reading directory block 16, phys 1581317
394166 0x0da08816-3cacd31a (24) ruby-net-telnet
518922 0x0db2dc78-251afc9e (20) xserver-xorg
[REMOVED] . . .
Entry #15: Hash 0xf07d620e, block 14
Reading directory block 14, phys 1585765
395617 0xf07d620e-e8bf0fde (16) orage
395676 0xf0d9ce20-b0dc591a (28) x11-session-utils
[REMOVED] . . .
```

395617: inode number 16: length of record orage: name of the folder	مثال: رقم مؤشر الفهرسة، طول التسجيلة، اسم المجلد (الدليل)

بعد التسجيلة الأخيرة dx_entry، (أي last node descriptor) بقية الكتلة ستكون مساحة مهملة slack space (أي مخصصة وبدون استخدام) عادة، هذه المساحة المهملة تتضمن بيانات من مدخلات دليل سابقة عندما كان الدليل يتناسب في كتلة واحدة.(أنظر للطرح) مثال على ذلك، مباشرة بعد نهاية التسجيلة الأخيرة dx_entry يمكنك رؤية جزء من مدخلة ثم المدخلة الأصلية للدليل الفرعي"alsa-utils". (نوع الملف 0x02) ثم مدخلة للدليل فرعي أخر يسمى "anacron" عند مؤشر الفهرس 0x606E1 ـ(394977). 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 المهملة قد تكون التسجيلة الوحيدة لوجودهم.

```
#debugfs -R "cat <123456>" /dev/sda1 | hexdump -C
     00 01 0 203 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 01 08 00 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 10 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 نسخ carving for 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 يملك الحقول التالية (كما تظهر في the first node descriptor) :

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

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

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

```
$ ls -1a ~
.
..
.bash_profile
.bashrc
mbox
public_html
tmp
```

	قيمة	حجم (بايت)	إزاحة (بايت)
مدخلة الدليل 1	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 3	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 5	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 7	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

	قيمة	حجم (بايت)	إزاحة (بايت)
مدخلة الدليل 0	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 2	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 4	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 6	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

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 الخصص 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
        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.........|
        0100
```

تحليل حقول ترويسة المديات extent header :

2 بايت تشير إلى الرقم السحري 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)ـ

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
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 00 02 00 |................|
0040 00 06 00 00 00 00 b2 14 00 00 08 00 00 94 02 00 |................|
0050 83 c5 14 00 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 |................|
0060 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 |................|
1000
```

2 بايت تشير إلى الرقم السحري 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 :

```
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 00 08 00 00 94 02 00 00 |................|
0050 83 c5 14 00 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 |................|
0060 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 |................|
1000

# filefrag -v sci-dictionary.pdf
Filesystem type is: ef53                    كتلة 2708 = 2707.003417969 = (4096 ÷ 11087886)
File size of sci-dictionary.pdf is 11087886 (2708 blocks of 4096 bytes)
 ext:     logical_offset:        physical_offset: length:   expected: flags:
   0:        0..       1:    1888971..    1888972:      2:
   1:        2..      31:    1313985..    1314014:     30:    1888973:
   2:       32..     127:    1314693..    1314788:     96:    1314015:
   3:      128..     511:    1349811..    1350194:    384:    1314789:
   4:      512..    2047:    1356288..    1357823:   1536:    1350195:
   5:     2048..    2707:    1361283..    1361942:    660:    1357824: last,eof
sci-dictionary.pdf: 6 extents found
```

1. الكتلة المنطقية : 0 ، عدد الكتل : 2، كتلة البداية : 1888971
2. الكتلة المنطقية : 2 ، عدد الكتل : 30، كتلة البداية : 1313985
3. الكتلة المنطقية : 32 ، عدد الكتل : 96، كتلة البداية : 1314693
4. الكتلة المنطقية : 128 ، عدد الكتل : 384، كتلة البداية : 1349811
5. الكتلة المنطقية : 512 ، عدد الكتل : 1536، كتلة البداية : 1356288
6. الكتلة المنطقية : 2048 ، عدد الكتل : 660، كتلة البداية : 1361283 (نهاية الملف)

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

# debugfs -R "dump_extents <548546>" /dev/sda1
Level Entries       Logical          Physical Length Flags
 0/ 1   1/  1     0 -  2707 2132319              2708
 1/ 1   1/  6     0 -     1 1888971 - 1888972      2
 1/ 1   2/  6     2 -    31 1313985 - 1314014     30
 1/ 1   3/  6    32 -   127 1314693 - 1314788     96
 1/ 1   4/  6   128 -   511 1349811 - 1350194    384
 1/ 1   5/  6   512 -  2047 1356288 - 1357823   1536
 1/ 1   6/  6  2048 -  2707 1361283 - 1361942    660
```

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 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. ملف sci-dictionary.pdf 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.

```
dd if=/dev/sda1 skip=2098698 bs=4096 count=506 | dd skip=6113 bs=256 count=1 | hexdump -Cv
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.........|
0100
```

```
blkcat /dev/sda1 2132319 | hexdump -C
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 00 08 00 00 94 02 00 00 |................|
0050 83 c5 14 00 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 |................|
0060 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 |................|
1000
```

قبل إلغاء تجزئة الملف

```
# debugfs -R "filefrag -v /home/xxx/Downloads/sci-dictionary.pdf" /dev/sda1
sci-dictionary.pdf has 2709 block(s), i_size is 11087886
 ext logical physical expected  length
   0       0  1888971              2
   1       2  1313985  1888973      30
   2      32  1314693  1314015      96
   3     128  1349811  1314789     384
   4     512  1356288  1350195    1536
   5    2048  1361283  1357824     660
sci-dictionary.pdf: 6 contiguous extents
```

إلغاء تجزئة الملف

```
# e4defrag -cv sci-dictionary.pdf
<File>
[ext 1]:   start 1888971:       logical 0: len 2
[ext 2]:   start 1313985:       logical 2: len 30
[ext 3]:   start 1314693:       logical 32:          len 96
[ext 4]:   start 1349811:       logical 128:         len 384
[ext 5]:   start 1356288:       logical 512:         len 1536
[ext 6]:   start 1361283:       logical 2048:        len 660
 Done.
# e4defrag sci-dictionary.pdf
ext4 defragmentation for sci-dictionary.pdf
[1/1]sci-dictionary.pdf:        100%       [ OK ]
Success:                        [1/1]
```

بعد إلغاء تجزئة الملف

```
# filefrag -v sci-dictionary.pdf
Filesystem type is: ef53
File size of sci-dictionary.pdf is 11087886 (2708 blocks of 4096 bytes)
 ext:     logical_offset:        physical_offset: length:   expected: flags:
   0:        0..    2707:     284932..    287639:   2708:             last,eof
sci-dictionary.pdf: 1 extent found
```

لفهم عمل هذه الآلية راجع كتيب "Outline of Ext4 File System & Ext4 Online Defragmentation Foresight"

121. ^ أ، ب، ت، ث، ج، الخصائص الممتدة EA وأذون قائمة التحكم بالنفاذ acl الممررة كخصائص ممتدة بين النواة ومساحة المستخدم، هي من الوسائل الفعالة والبسيطة في حماية أنظمة لينكس الداخلية، إلى جانب تطبيقات الأخرى مثل SELinux. وللتعامل مع الخصائص الممتدة، أنت في حاجة إلى فهم استخدام chattr و lsattr، الأداة الأولى تقوم بتعيين وإزالة الخصائص الممتدة (باستثناء بعض الخصائص)، والثانية (مثل ls) تقوم بعرض الخصائص المرتبطة بالملف المحدد. 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" كما في المثال: chattr +i some_special_file. بالمناسبة الأمر ls لن يعرض هذه الخاصية الموجودة في الملف، فقط أداة lsattr تستطيع عرض الخاصية. ولإزالة أية خاصية ممتدة في الملف، نستخدم خيار "S" كما في المثال: chattr -S some_file

1. 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 الوصل user_xattr
2. أيضا استخدام الخيارات المبدئية/الاعتيادية للوصل بدلا من استخدام مدخلة ملف /etc/fstab سيكون مفيد مع الأجهزة الأقراص الخارجية.
3. مبدئيا هذه الخيارات ستكون موجودة، عند إنشاء نظم الملفات ext2/3/4
4. يمكنك تعين وعرض الخصائص الممتدة بواسطة setfattr و getfattr.
5. في تعيين أذون ACLs على الملفات، نستخدم setfacl و getfacl. عموما حتى تستطيع استخدام هذه الأدوات مع ACLs، يجب وصل نظام الملفات عن طريق خيار الوصل acl.

1. `UUID=661ab9f1-c381-4962-bcfc-0b5e2aab1ce9 /home ext4 defaults,user_xattr,acl  1 2` (etc/fstab)
2. `# tune2fs -l /dev/sda1 | grep "Default mount options:"`
   `Default mount options:    user_xattr acl`
3. `# cat /etc/mke2fs.conf | grep acl`
   `default_mntopts = acl,user_xattr`
4. `$ 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"`
5. `# tune2fs -o acl /dev/sda1` (تمكين acl كخيار وصل)

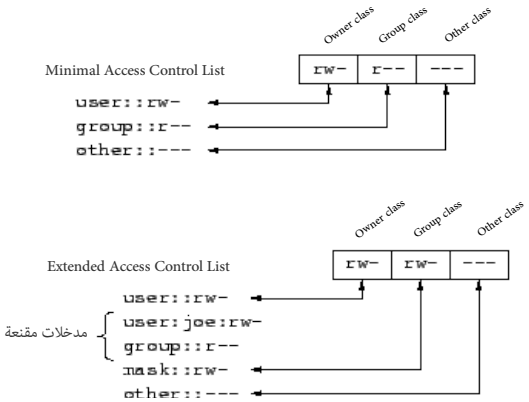
تقليديا، الملفات تملك ثلاثة تصاريح أو أذون مختلفة: قراءة، وكتابة، وتنفيذ rwx، من أجل ثلاثة أصناف أومجموعات مختلفة هي المستخدم user، والمجموعة group، والآخرون other. يمكن للمدير إعطاء أشخاص معينين صلاحية الكتابة إلى ملف معين، عبر إنشاء مجموعة تملك تلك الصلاحية ثم ضم أولئك الأشخاص إلى المجموعة. لكن مع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

خرج getfacl مثال 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 تقترن بحقول أصناف user، و group، و other في بتات أذون أنماط الملفات. هذه الثلاثة تسمى مدخلات أساسية أو مدخلات دنيا (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 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

ACL نوع مدخلة		صيغة	في هذا المثال: المستخدم john يملك الأذون النافذة r-x ؛ لكنه لا يملك حق الكتابة إلى الملف، لأن w لم يعين في القناع.		
Owner	مالك	user::rwx			
Named user	مستخدم مسمى	user:name:rwx			
Owning group	مجموعة	group::rwx			
Named group	مجموعة مسماة	group:name:rwx	حقل		أذون
Mask	قناع	mask::rwx	named user	مستخدم مسمى	user:john:rwx
Others	آخرون	other::rwx	mask	قناع	mask::r-x

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 الغاء 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 خابية كتابة write cache جهاز القرص سوف تكتب أخيرا إلى أطباق القرص. على شرط أن لا تحدث حالة starvation condition 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 (E7h) و (EAh) التي ستجعل القرص يكمل كتابة البيانات من الخابية، وسيعود القرص في حالة جيدة بعد كتابة بيانات write cache إلى القرص الوسيط. أيضا، flushing يمكن أن يستهل على الأقل مع بعض الأقراص عن طريق 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 حجم 1 جيجابايت.
125. ^ الصوان أو الذاكرة الوسيطة Buffer هو مكان مؤقت في الذاكرة حيث يتم فيه تخزين البيانات حين تنقّل من مكان إلى آخر. كما ويتعلق تركيبه حسب الوظيفة التي يشغلها. عادة البيانات تخزن قي Buffer بالشكل الذي تأتى منه كأي وحدة إدخال كلوحة المفاتيح أو قبل أن ترسل إلى أي وحدة إخراج كالطابعة. لكن يمكن أستخدامها أثناء نقل البيانات بين العمليات التي تجرى داخل الحاسوب.
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:

```
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 block **descriptor** تسمى أوسمة توصيف (أو لواحق !) **descriptor** tags، (هياكل صغرى) journal descriptor block تخزن تعيين كل metadata buffer في journal إلى موقعها الفعلي على القرص في شكل tags

130. ^ أدلة يونكس / لينكس

شرح	رمز	اسم الدليل / الحاوية	
أعلى دليل (عقدة) في أية بنية ملفات في يونكس	/	دليل جذر	Root Directory
دليل المستخدم !	~	دليل منزل	Home Directory
الموقع الاعتيادي عند العمل مع الملفات	.	دليل حالي	Current Directory
دليل مباشرة فوق الدليل الحالي current directory	..	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 (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:

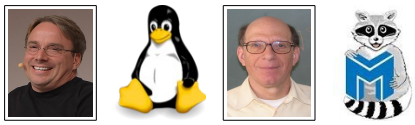
أداوت القرص	Disk Tools	فئة البيانات الوصفية	Metadata Category
أدوات نظام وحدة التخزين	Volume System Tools	فئة أسماء الملفات	File Name Category
أدوات نظام الملفات	File System Tools	فئة التطبيقات	Application Category
فئة نظام الملفات	File System Category	فئة متعددة	Multiple Category
فئة المحتوى	Content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

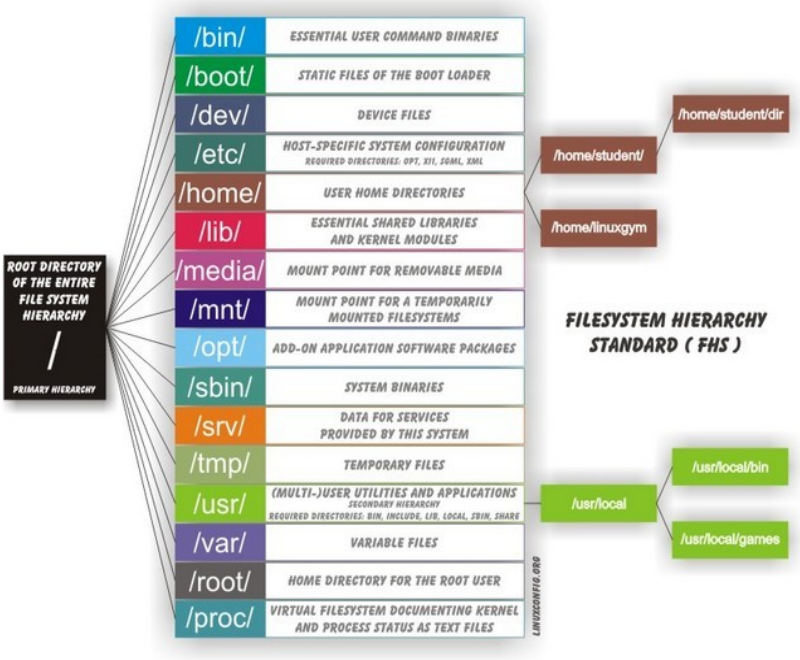
وظيفة		أداة TSK
عرض محتويات كتلة (عرض وحدة بيانات نظام الملفات في صورة القرص)	Views the contents of a block.	Blkcat
سرد أو خرج (طرح) وحدات بيانات نظام الملفات (أي الكتل)	List or output file system data units	blkls
التحويل بين أرقام وحدات القرص بدون تخصيص وأرقام وحدات القرص الاعتيادية	Converts between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

132. ^ مؤشر الفهرسة المعزول orphaned inode هو ذلك inode الذي ليس له رابط 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. خرج lsof سوف يعرض ذلك

133. ^ جذور نظام الملفات

- اندرو ستيوارت تانينباوم هو من كتب نظام مينيكس لغرض تدريس تصميم أنظمة التشغيل وعمل النواة، وتمّ طرحه كنظام مفتوح المصدر في أبريل 2000 تحت رخصة BSD.
- نظام ملفات مينيكس أول نظام ملفات في لينكس، يملك بنية تحاكي UFS من 6 عناصر Boot sector, superblock, inode bitmap, zone bitmap, inodes area, data area.
- 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.
- ريمي كارد صمم ext لتخطى حدود نظام ملفات مينيكس. ويغلب على ext بنية البيانات الوصفية المستوحاة من نظام ملفات يونكس UFS/FFS.
- ريمي كارد صمم أيضا ext2 المستخدم في نواة لينكس كبديل لنظام ext.
- ext3 المزود بقيد 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





التسلسل الهرمي القياسي لنظام الملفات

134. ^ حواجز الكتابة Write Barriers:

آلية في النواة تستخدم للضمان تنظيم ordered وكتابة written البيانات الوصفية للنظام الملفات على جهاز التخزين المستقر persistent storage (أي الغير متطاير) حتى في حالة انقطاع الكهرباء عن أجهزة التخزين ذات خابية الكتابة المتطايرة volatile write caches. في حال تمكين حواجز الكتابة 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 دئما ()fsync أوتعمل على إنشاء وحذف الملفات الصغيرة الكثيرة، ستعمل أبطأ.

135. ^ نواة لينكس تدعم أنظمة الملفات (الفيزيائية) التالية: Amiga FFS، BFS، cramfs، ext2، ext3،ext4، F2FS، FAT، HFS، HPFS، ISO 9660، JFFS، JFFS2، JFS، MINIX fs، NSS، OCFS ، UDF، UFS، XFS، ZFS، ReiserFS، Reiser4، Btrfs، Tux3، ZFS (دعم القراءة فقط) و BeFS، EFS، FreeVxFS و System V FS (دعم الكتابة ؟) و+HFS (دعم الكتابة محدود/فقط مع قيد الحوادث الخالي)، و Acorn ADFS (دعم الكتابة تجريبي) و QNX4 FS (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. نواة لنكس تدعم أنظمة الملفات الشبكية التالية: NFS، AFS، CIFS، Coda، 9P، Ceph

مراجع

1. ^ تخطيط قرص Ext4 (المرجع الأول لهذه الكتيب)
2. ^ نظام ملفات 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 مجامع 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. ^ نظام الملفات الممتدة / المحسن Ext4
21. ^ خصائص وأذون acl و ext3 و ext4 و fsck، من موقع archlinux
22. ^ مواضيع من موقع Linux.org
23. ^ موقع linuxconfig.org
24. ^ المقارنة بين أنوية أنظمة التشغيل
25. ^ محاولة استعادة الدليل المحذوف في ext4 ، موقع opensuse
26. ^ موضوع Ext4 on SSD Intel X25-M من marc.info
27. ^ موضوع Reserved GDT blocks من redhat.com
28. ^ مقالات عن ext4 من computer-forensics.sans.org
29. ^ معلومات من stackoverflow.com
30. ^ تحليل بنية نظام ملفات ext4 من موقع programering.com
31. ^ موضوع Undeletion
32. ^ موضوع Ext2-Ext3-Ext4 Attributes و Ext2/Ext3/Ext4 File System من موقع softpanorama.org
33. ^ next3-utils من أمير (المشروع ميت منذ أكتوبر 2013 ؟) Amir's ext4 snapshot work، رقع EXT4 snapshot patches في ithub.com و article.gmane.org
34. ^ موضوع "A Minimum Complete Tutorial of Linux ext4 File System" من مدونة metebalci موقع medium.com
35. ^ 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)
36. ^ موقع Flylib.com

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

تَنْبِيهٌ

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!ـ 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

جمادى الأول / يناير/كانون الثّاني 2019

تمت بحمد الله