INSTITUTO DE MATEMÁTICA E ESTATÍSTICA - USP

MAC0499 - Trabalho de Formatura Supervisionado

Desenvolvimento de um programa para Windows 7 compatível com os Sistemas de Arquivos Ext2, Ext3 e Ext4

Felipe Simionato Solferini

Orientação: Professor Marco Dimas Gubitoso

10 de fevereiro de 2013

Sumário

1 Introdução **1**

2 Sistemas de Arquivos **2**
2.1 Estruturas do disco . . . 2
2.1.1 Partição . . . 2
2.1.2 MBR . . . 2
2.1.3 EBR . . . 3
2.1.4 Endereçamento LBA . . . 4
2.1.5 Endereçamento CHS . . . 4
2.1.6 Tabela de Partição . . . 4
2.1.7 Tipos de Entrada Extended . . . 4
2.2 Estruturas dos Sistemas de Arquivos Ext . . . 5
2.2.1 Superbloco . . . 5
2.2.2 Journaling . . . 5
2.2.3 Blocos . . . 5
2.2.4 Grupos de blocos . . . 5
2.2.5 Descritores de Grupo de blocos . . . 6
2.2.6 Mapa de bits . . . 6
2.2.7 Inode . . . 6
2.2.8 Endereçamento direto ou indireto de blocos . . . 7
2.2.9 *Extent Tree* . . . 8

3 O Extended Filesystem **9**
3.1 Ext2 . . . 9
3.1.1 Superbloco . . . 10
3.1.2 Tabela de Descritores de Grupo . . . 10
3.1.3 Tabela de Inodes . . . 11
3.1.4 Entrada de diretório . . . 12
3.2 Ext3 . . . 13
3.3 Ext4 . . . 14

4 Arquitetura e implementação **16**
4.0.1 Identificação de discos . . . 16
4.0.2 Identificação das partições Ext . . . 17
4.0.3 Recuperação das informações das partições Ext . . . 17
4.0.4 Interação com o usuário . . . 18

5 Testes **19**

6 Conclusão **23**

7 Parte subjetiva **24**
7.1 Desafios e frustrações . . . 24
7.2 Disciplinas cursadas . . . 25
7.3 Comentários . . . 26
7.4 Próximos passos . . . 27

8 Apêndice **28**
8.1 Entradas do Superbloco 28
8.1.1 Flags e Identificadores do Superbloco 31
8.2 Descritores de grupo de blocos 33
8.3 Tabela de inode 34

1 Introdução

Com o advento da World Wide Web, o Sistema Operacional Linux foi largamente difundido pelo mundo afora, assim como os ideais de liberdade do Software Livre. No princípio o Linux era usado apenas por experts da área da computação, mas com o passar dos anos foi ganhando mais e mais adeptos com a popularização dos ambientes gráficos (Gnome, KDE...).

Este trabalho tem como objetivo construir um programa que possibilite aos usuários de Sistemas Operacionais Linux e Windows, em dual boot, acessarem seus arquivos do Linux enquanto estão no ambiente Windows. Com isso pretendo incentivar os calouros dos cursos da área de informática a migrarem gradativamente para o Linux.

No decorrer do trabalho, apresentarei os principais conceitos que um aluno do quarto ano de um curso de Ciência da Computação precisa saber para conseguir acompanhar a monografia sem dificuldades.

Os capítulos 2 e 3 apresentam uma descrição dos elementos que compõem os sistemas de arquivos Ext e como eles estão distribuídos. Eles trazem um aprofundamento do que aprendemos no curso de Sistemas Operacionais.

Nos capítulos 4 e 5, exponho a arquitetura montada para estruturar o programa, objetivo principal deste trabalho, e apresento os testes realizados que visam garantir a integridade dos dados extraídos.

No apêndice eu apresento algumas informações que eu julgo relevantes para a compreensão do meu trabalho, ou que representam um aprofundamento no tema, mas que são dispensáveis à maioria dos alunos e professores. Esse conteúdo foi colocado à parte para deixar a leitura do trabalho menos monótona.

Durante toda a monografia, discos serão referenciados seguindo uma abstração lógica, ou seja, serão tratados como uma longa fita onde os bits são armazenados sequencialmente. Essa é a visão que o sistema operacional tem dos dados, mas não é a forma que eles realmente são armazenados no disco.

2 Sistemas de Arquivos

Sistema de Arquivos é o nome dado à entidade responsável por armazenar os arquivos e diretórios. Ele ocupa inteiramente uma partição do disco. Existem diversas maneiras de armazenar arquivos em uma partição usando diferentes métodos de indexação e alocação de espaço. As maneiras relevantes a esse trabalho serão apresentadas a seguir.

2.1 Estruturas do disco

2.1.1 Partição

Uma partição pode ser entendida como a divisão de um disco em "subdiscos". Os subdiscos mantém a estrutura básica de inicialização, que ocupa os primeiros 512 bytes do disco, chamada de MBR no disco físico e VBR no subdisco. Isso cria uma abstração para os Sistemas de Arquivos, que não precisam se preocupar se o disco está particionado ou não.

2.1.2 MBR

Nos primeiros 512 bytes do disco, está localizado o Registro Mestre de Inicialização, ou MBR (Master Boot Record). Nele encontramos informações sobre o processo de inicialização (boot), além de uma tabela com 4 entradas, cada uma contendo informações sobre uma partição diferente do disco. Quando essa estrutura está no começo de uma partição, é chamada de Registro de Volume de Inicialização, ou VBR (Volume Boot Record). No final do MBR (ou VBR) encontramos o número 0xAA55. Esse número nada mais é do que uma forma de termos certeza de que aquilo que foi lido é um registro de inicialização.

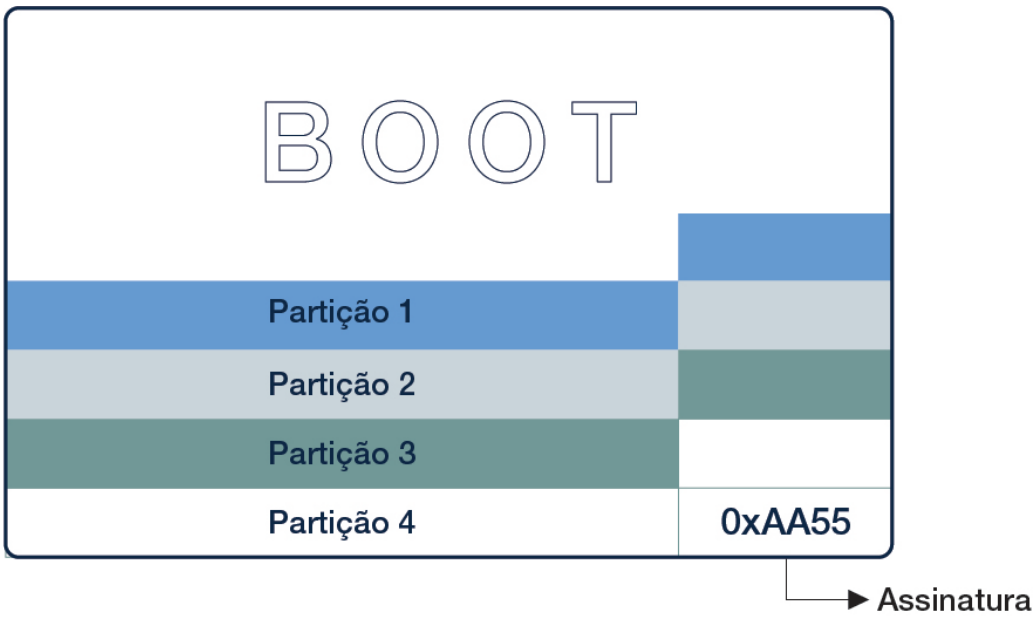

Figura 1: MBR

2.1.3 EBR

Devido à limitação de quatro entradas na tabela de partição da MBR, foi criado o Registro Estendido de Inicialização, ou EBR (Extended Boot Record).

A estrutura do EBR é similar à do MBR, mas apenas a primeira e a segunda entrada são utilizadas, a primeira como um apontador para a partição e a segunda como um ponteiro para o próximo EBR, formando assim uma lista ligada.

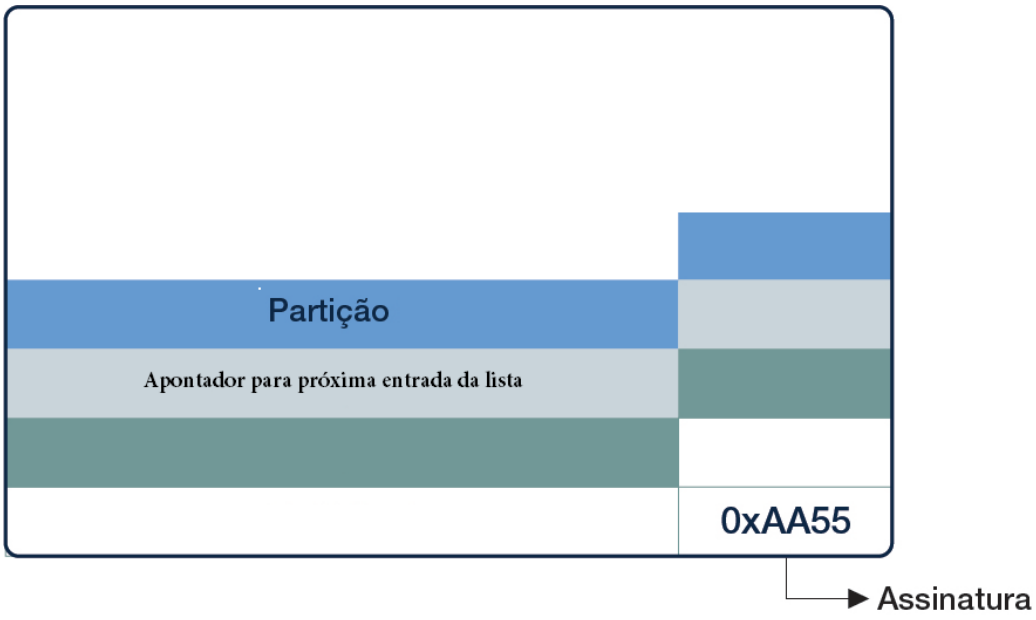

Figura 2: EBR

Quando temos mais do que 4 partições no disco, a última entrada do MBR é usada como um apontador para um EBR, iniciando assim a lista ligada.

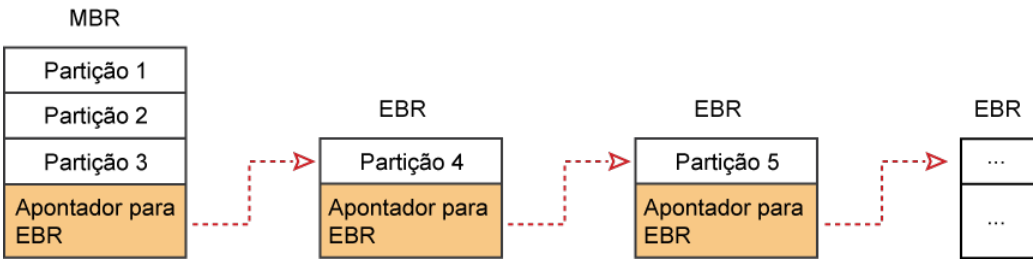

Figura 3: Lista ligada formada pelos EBRs

2.1.4 Endereçamento LBA

Endereçamento de Bloco Lógico ou Logical Block Addressing (LBA) é uma forma de endereçamento que utiliza a posição lógica que o bloco (ou setor) ocupa no disco em vez de usar o endereçamento CHS.

2.1.5 Endereçamento CHS

CHS (Cylinders, Head, Sectors ou cilindros, cabeçotes e setores) é uma forma de endereçamento obsoleta que utiliza a geometria do disco para localizar a posição das partições. Ela caiu em desuso com o aumento do tamanho dos discos, já que é capaz de representar, supondo um setor de 512 bytes, apenas 8.4GB.

2.1.6 Tabela de Partição

A Tabela de Partição pode ser entendida como um vetor de 4 posições em que cada entrada possui informações sobre a localização, tamanho e tipo das partições.

Início	**Tamanho**	**Descrição**
0x0	1 byte	É inicializável? (0x80 -> Sim; 0x00 -> Não)
0x1	3 bytes	Setor inicial (em CHS)
0x4	1 byte	Tipo da partição (ver lista)
0x5	3 bytes	Setor final (em CHS)
0x8	4 bytes	Setores relativos (em setores)
0xC	4 bytes	Tamanho (em setores)

2.1.7 Tipos de Entrada Extended

As entradas que apontam para os EBRs podem ser de 2 tipos, que variam de acordo com a forma de indexação. As do tipo 0x5 usam a posição relativa ao EBR para serem localizadas. Já as do tipo 0xf usam a posição absoluta, ou seja, a partir do início do disco.

2.2 Estruturas dos Sistemas de Arquivos Ext

2.2.1 Superbloco

O Superbloco pode ser entendido como um registro geral do sistema de arquivos. Nele encontramos informações sobre blocos e inodes, estruturação dos grupos de blocos, ocorrência de erros, entre outros. Ele nos fornece as condições necessárias para encontrarmos as outras estruturas do sistema de arquivos.

2.2.2 Journaling

Journaling, ou Registro, é um mecanismo de recuperação de erros criado para reduzir o tempo de verificação da integridade do sistema de arquivos. A verificação pode ocorrer de tempos em tempos, como uma medida de prevenção de erros, ou depois de um desligamento inesperado, para retornar o sistema de arquivos a um estado consistente. Também pode acontecer quando o usuário requisitar.

O sistema de arquivos mantem um registro das alterações feitas nos arquivos. Com isso, em vez de verificar todos os arquivos do disco, basta verificar a integridade dos últimos arquivos alterados. O controle é feito usando transações, como costuma ocorrer em sistemas de bancos de dados.

2.2.3 Blocos

Bloco é o nome dado à unidade mínima de alocação de um sistema de arquivos. Blocos são usados para guardar todo tipo de informação e as principais estruturas têm seu tamanho dado em blocos, ou seja, elas ocupam um espaço que é múltiplo do tamanho do bloco. São elas: Superbloco, Tabela de Descritores de Grupo, Mapa de Bits dos blocos de dados, mapa de bits dos inodes, Tabela de inodes e blocos de dados.

Suponhamos que num dado sistema de arquivos o bloco tenha 1KB e a tabela de inodes de um determinado grupo de blocos esteja localizada no 1024000º byte do sistema de arquivos. Ao invés de dizermos que ele se encontra na posição 1024000, caso estivéssemos indexando por bytes, dizemos que ele está na posição 1000 (1024000 bytes/ 1KB). Com isso aumentamos o tamanho máximo do sistema de arquivos.

Essa estruturação acaba forçando os diretórios e arquivos a ocuparem um espaço que seja múltiplo do tamanho de um bloco, apesar de o seu tamanho. Ou seja, para o usuário e para o sistema operacional, o tamanho é o mesmo.

2.2.4 Grupos de blocos

Grupos de blocos são agrupamentos usados para aumentar a chance de um arquivo ser escrito em uma sequência contígua de blocos. Isso evita a fragmentação dos arquivos e consequentemente reduz o movimento da cabeça de leitura do disco. Cada grupo de blocos possui uma cópia do superbloco, descritores de grupo, um mapa de bits para seus blocos, um mapa de bits para seus inodes, uma tabela de inodes e blocos de dados. É como se o sistema de arquivos repassasse as suas obrigações para uma entidade menor.

As estruturas que têm posição definida são o superbloco e a tabela de descritores de grupo. Todas as outras estruturas não tem uma posição fixa, ou seja, elas flutuam.

2.2.5 Descritores de Grupo de blocos

Os descritores de grupo de blocos são responsáveis pelo registro das informações referentes às estruturas internas dos grupos de blocos. Cada descritor de grupo está associado a um grupo de blocos e nos diz onde está localizada a tabela de inodes do grupo de blocos.

2.2.6 Mapa de bits

Um bitmap ou mapa de bits é um vetor em que cada bit representa uma posição do vetor e está associado a um elemento de um conjunto. Ele pode ser usado para marcar quais blocos ou inodes estão em uso em um sistema de arquivos.

2.2.7 Inode

Um index node ou inode, como é conhecido, é uma estrutura de dados criada com o propósito de armazenar um arquivo (ou diretório) no disco. Informações como data de criação, tamanho, tipo de indexação de blocos e ponteiros para os blocos de dados são encontradas nele. O nome do arquivo fica armazenado na entrada de diretório do seu inode pai.

Os blocos podem ser indexados usando endereçamento direto ou indireto de blocos, ou uma *Extent Tree*.

2.2.8 Endereçamento direto ou indireto de blocos

O endereçamento direto ou indireto de blocos funciona de uma maneira bem simples. Existem quatro tipos de apontadores:

I Direto: aponta para um bloco de dados;

II Indireto simples: aponta para um vetor de blocos de dados;

III Indireto duplo: aponta para um vetor de apontadores do tipo II;

IV Indireto triplo: aponta para um vetor de apontadores do tipo III.

No caso dos Exts, os inodes possuem 12 apontadores diretos e 1 apontador para cada um dos outros. Para indexar novos blocos, são sempre usadas as estruturas menos complexas, ou seja, I,II,III e IV, em ordem de prioridade.

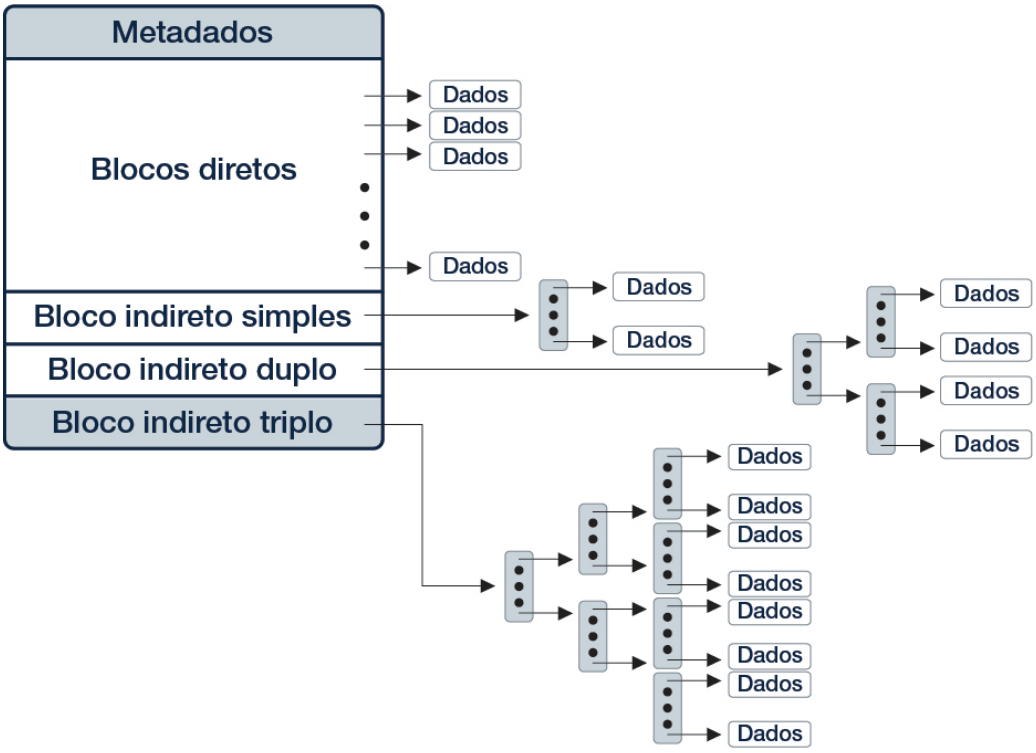

Figura 4: Indexação direta e indireta de blocos

2.2.9 *Extent Tree*

Extent Tree é uma estrutura de indexação de blocos criada para reduzir a fragmentação dos arquivos. As folhas da árvore armazenam um apontador para o primeiro bloco de uma sequência de blocos, o tamanho da sequência e qual posição o primeiro bloco ocupa no arquivo.

Toda entrada da árvore possui uma estrutura comum, chamada cabeçalho (header). O cabeçalho nos diz, principalmente, quantos nós a entrada armazena e qual é a altura dessa entrada.

Um nó cuja altura não é zero é chamado de nó interno e aponta para um bloco que contém um cabeçalho seguido de vários nós. Caso a altura seja zero, o nó é uma folha da árvore.

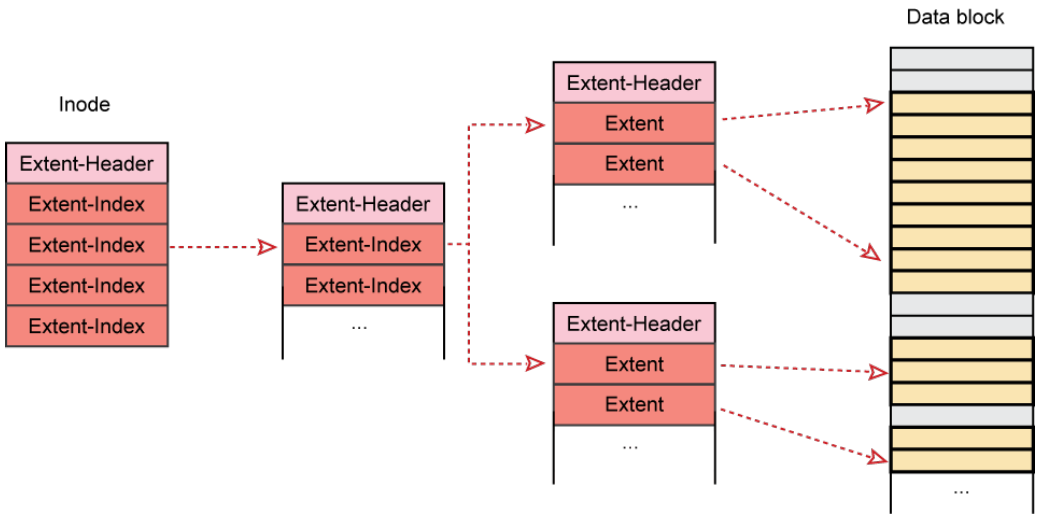

Figura 5: Indexação usando uma *Extent Tree*

3 O Extended Filesystem

O Ext FS foi criado em Abril de 1992 para substituir o defasado Minix FS que era o sistema de arquivos usado pelo Linux na época. Esse novo sistema de arquivo tinha um tamanho máximo de 2GB, comportando arquivos de no máximo 2GB com nomes de até 255 caracteres. Apesar do grande avanço, o Ext FS tinha um problema muito incômodo. Ele usava uma lista ligada não ordenada para controlar os blocos e inodes livres. Isso fazia com que quanto mais o sistema de arquivos fosse usado, mais ele ficava fragmentado.

Foi em Janeiro de 1993, para solucionar esses problemas que foi lançado o Ext2 FS. Ele foi projetado para ser um sistema de arquivos robusto, que reduziria a fragmentação e o risco de perda de dados.

As tabelas apresentadas no decorrer desse capitulo são versões resumidas contendo as informações necessárias para a leitura, o que já é suficiente para a extração dos arquivos, objetivo desse trabalho. A versão completa das tabelas está disponível no apêndice.

3.1 Ext2

O Second Extended Filesystem, ou Segundo Sistema de Arquivos Estendido, surgiu do esforço conjunto dos pesquisadores Rémy Card, Theodore Ts'o e Stephen C. Tweedie. Um pouco da história dessa criação pode ser vista no paper Design and Implementation of the Second Extended Filesystem[ALL].

O Ext2 possuí uma margem de 1024 bytes, espaço esse reservado para as estruturas de inicialização (MBR ou VBR). Após essa margem o sistema de arquivos está dividido em grupos de blocos de mesmo tamanho, com exceção do último que pode ser menor, já que acomoda os blocos resultantes do resto da divisão do tamanho do sistema de arquivos pelo tamanho de um grupo de blocos.

Em cada grupo de blocos encontramos as seguintes estruturas: Superbloco, Tabela de descritores de grupo, Mapa de bits de blocos, Mapa de bits de inodes, Tabela de inodes e blocos de dados.

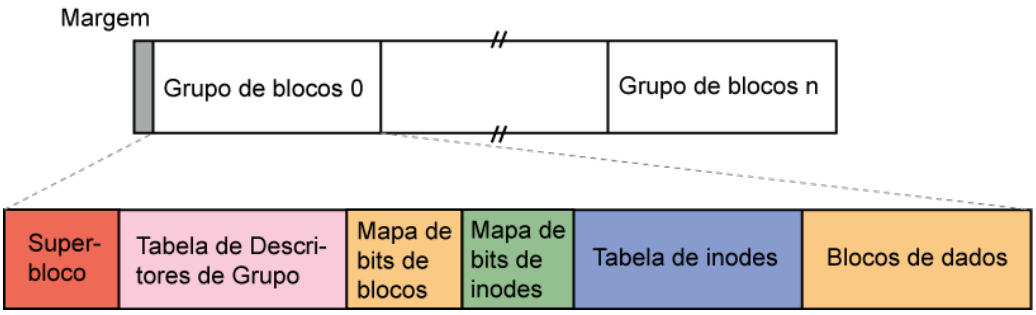

Figura 6: Extrutura de um Ext

3.1.1 Superbloco

O Superbloco ocupa 1024 bytes e possuí uma cópia de redundância em cada um dos descritores de grupo. Por questões de desempenho, muitas vezes apenas o Superbloco do Grupo de Blocos 0 é usado e atualizado.

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x00	0x03	4	Quantidade de inodes no Sistema de Arquivos
0x04	0x07	4	Quantidade de blocos no Sistema de Arquivos
0x18	0x1B	4	Log2 do tamanho de um bloco, em KB.
0x20	0x23	4	Quantidade de blocos em cada Grupo de Blocos
0x28	0x2B	4	Quantidade de inodes em cada Grupo de Blocos
0x38	0x39	2	Assinatura do Ext (0xEF53)
0x4C	0x4F	4	Porção mais significante da Versão

Se a versão for maior do que 1

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x58	0x59	2	Tamanho do inode

3.1.2 Tabela de Descritores de Grupo

Assim como o Superbloco, a tabela de Descritores de Grupo também possui uma cópia de redundância em cada um dos descritores de grupo. Essas redundâncias ocorrem para garantir que o sistema consiga se recuperar caso alguma das informações seja corrompida. Cada descritor de grupo ocupa 32 bytes. Para encontrarmos o descritor de grupo referente ao N-ésimo grupo, basta procurarmos na posição N*32 (em bytes) a partir do início da tabela.

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x08	0x0B	4	Posição inicial da tabela de inodes do Grupo de Blocos

3.1.3 Tabela de Inodes

Dado um inode x, para determinarmos a qual grupo de blocos N está a tabela a qual ele pertence, basta realizarmos a seguinte operação:

$$N = (x-1)/INODES\_POR\_GRUPO$$

Para encontrarmos o índice i relativo do inode x, na tabela de inodes, fazemos:

$$i = (x-1)\%INODES\_POR\_GRUPO$$

Um inode, por padrão, ocupa 128 bytes, mas esse valor pode variar, se definido explicitamente. Encontramos sua posição no grupo de blocos N, na posição i*tamanho_do_inode (em bytes).

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x00	0x01	2	Tipo e permissões
0x28	0x2B	4	Apontador direto 0
0x2C	0x2F	4	Apontador direto 1
0x30	0x33	4	Apontador direto 2
0x34	0x37	4	Apontador direto 3
0x38	0x3B	4	Apontador direto 4
0x3C	0x3F	4	Apontador direto 5
0x40	0x43	4	Apontador direto 6
0x44	0x47	4	Apontador direto 7
0x48	0x4B	4	Apontador direto 8
0x4C	0x4F	4	Apontador direto 9
0x50	0x53	4	Apontador direto 10
0x54	0x57	4	Apontador direto 11
0x58	0x5B	4	Apontador indireto simples
0x5C	0x5F	4	Apontador indireto duplo
0x60	0x63	4	Apontador indireto triplo

Tipos:

- 0x1000 FIFO
- 0x2000 Dispositivo de caractere
- 0x4000 Diretório
- 0x6000 Dispositivo de bloco
- 0x8000 Arquivo comum
- 0xA000 Link simbólico
- 0xC000 Socket Unix

3.1.4 Entrada de diretório

Um diretório é um arquivo comum que, ao invés dos dados do arquivo, possui entradas de diretório em seus blocos. As entradas de diretório terminam no final do arquivo. Entradas não utilizadas são identificadas pelo valor 0 no campo de inode.

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x00	0x03	4	Inode
0x04	0x05	2	Tamanho desta entrada de diretório
0x06	0x06	1	Tamanho do nome do arquivo
0x07	0x07	1	Tipo
0x08	0x08+N-1	N	Nome

Tipos:

- 0x00 Tipo desconhecido
- 0x01 Arquivo comum
- 0x02 Diretório
- 0x03 Dispositivo de caracter
- 0x04 Dispositivo de bloco
- 0x05 FIFO
- 0x06 Socket Unix
- 0x07 Link simbólico

3.2 Ext3

O Ext2 representou um marco na história da computação e certamente teve um importante papel na difusão do Linux. Apesar de todas as suas qualidades, a recuperação de erros era um problema bem incomodo. Ela tomava uma quantidade de tempo desnecessária, uma vez que todos os arquivos e diretórios do sistema de arquivos eram verificados. Com o crescimento do tamanho dos discos os desenvolvedores sabiam que isso se tornaria impraticável num futuro próximo. Esse foi o principal motivo para o desenvolvimento do Ext3, como relatado por Stephen C. Tweedie no paper Journaling the Linux ext2fs Filesystem[SCT1].
O Ext3 foi construído assumindo duas premissas:

- Deveria usar *Journaling* para o controle de suas atividades;
- Deveria ser compatível, na medida do possível, com o Ext2.

Com o sistema de *Journaling*, são verificadas apenas as últimas transações realizadas pelo sistema.

Para fins de compatibilidade com o Ext2, o Ext3 consiste essencialmente no Ext2 com o *journaling*. Particularmente, para convertermos de Ext2 para Ext3, precisamos, *apenas* adicionar o journaling: o restante da estrutura se mantém.

3.3 Ext4

O Ext4 apresentou mudanças substanciais na forma como os arquivos são dispostos. A própria arquitetura dessas mudanças força os arquivos a não se fragmentarem. O recurso de Extent Trees foi escrito de uma forma que favorece o uso de blocos contíguos, acelerando a velocidade de leitura do arquivo e dificultando a fragmentação, já que ela ocorre quando o arquivo fica espalhado pelo disco. Os grupos de blocos flexíveis fazem com que várias tabelas de inode e tabelas de descritores de grupo fiquem escritas no mesmo bloco, deixando assim mais espaço para alocação contígua. Note que o uso de Extent Trees não exclui a possibilidade de alguns inodes usarem a indexação padrão.

Outro recurso que vale a pena ser citado é o de Superblocos esparsos. Com esse recurso, as cópias de redundância do superbloco e da tabela de descritores de grupo são mantidas apenas no bloco 0, ou nos blocos que são potências de 3, 5 ou 7. Isso, em conjunto com os grupos de blocos flexíveis, faz com que o arquivo não precise ser fragmentado, caso ele precise ocupar mais de um grupo de blocos.

O Ext2 e o Ext3 são casos particulares do Ext4. Eles podem ser entendidos como um sistema de arquivos Ext4 com muitos dos recursos extras desativados. Isso permite que eles sejam lidos como Ext4, sem problemas.

A única informação do superbloco relevante à leitura do Ext4 usada neste trabalho, que é diferente das contidas nas tabelas do ext2, é o tamanho dos descritores de grupo:

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0xFE	0xFF	2	Tamanho dos descritores de grupo

O que realmente difere o Ext4 dos seus antecessores, quando pensamos em leitura, é a existência das Extent Trees. Usando essa estrutura, podemos escrever os blocos de um arquivo de, por exemplo, 1000 blocos usando apenas um nó de uma Extent Tree, ao invés de gastar toda a tabela de inodes, além de um ou mais blocos no endereçamento indireto simples. Ou seja, gastaremos apenas 24 bytes já reservados no inode (12 bytes para o cabeçalho e 12 bytes para o nó) ao invés dos aproximadamente 4 KB gastos pelo método antigo.

Cabeçalho da Extent Tree:

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x00	0x01	2	bytes Assinatura do cabeçalho (0xF30A)
0x02	0x03	2	bytes Quantidade de nós dessa entrada
0x04	0x05	2	bytes Quantidade máxima de nós da entrada
0x06	0x07	2	bytes Altura dessa entrada

Nós internos:

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x04	0x07	4 bytes	32 bits menos significativos do filho do nó
0x08	0x09	2 bytes	16 bits mais significativos do filho do nó

Folhas:

Posição inicial	**Posição final**	**Tamanho (em bytes)**	**Descrição**
0x00	0x03	4 bytes	Primeiro bloco da sequência contígua
0x04	0x05	2 bytes	Quantidade de blocos da sequência
0x06	0x07	2 bytes	16 bits mais significativos do bloco
0x08	0x0B	4 bytes	32 bits menos significativos do bloco

Com as informações disponibilizadas neste capitulo já temos condições suficientes para extrairmos quaisquer arquivos de um sistema de arquivos Ext.

4 Arquitetura e implementação

O extrator de arquivos foi desenvolvido usando apenas as bibliotecas mais básicas da linguagem C. Para a manipulação dos discos e arquivos foram usadas as bibliotecas de leitura e controle de entrada e saída de dispositivos da Win32 API (<Windows> e <WinIoCtl>). Toda a lógica do programa foi desenvolvida usando as bibliotecas C Strings (<string>), C Standard Input and Output (<stdio>) e C Standard General Utilities (<stdlib>). O processo de codificação foi feito com o auxílio do editor de textos Notepad++ (http://notepad-plus-plus.org/). Para o processo de compilação dos códigos fonte foi usado o GNU Make em conjunto com o compilador C da GNU Compiler Collection.
O programa é dividido em 4 etapas:

- Identificação dos discos
- Identificação das partições Ext
- Recuperação das informações das partições Ext
- Interação com o usuário

Devido a questões de segurança, o programa precisa de privilégios de Administrador para funcionar.

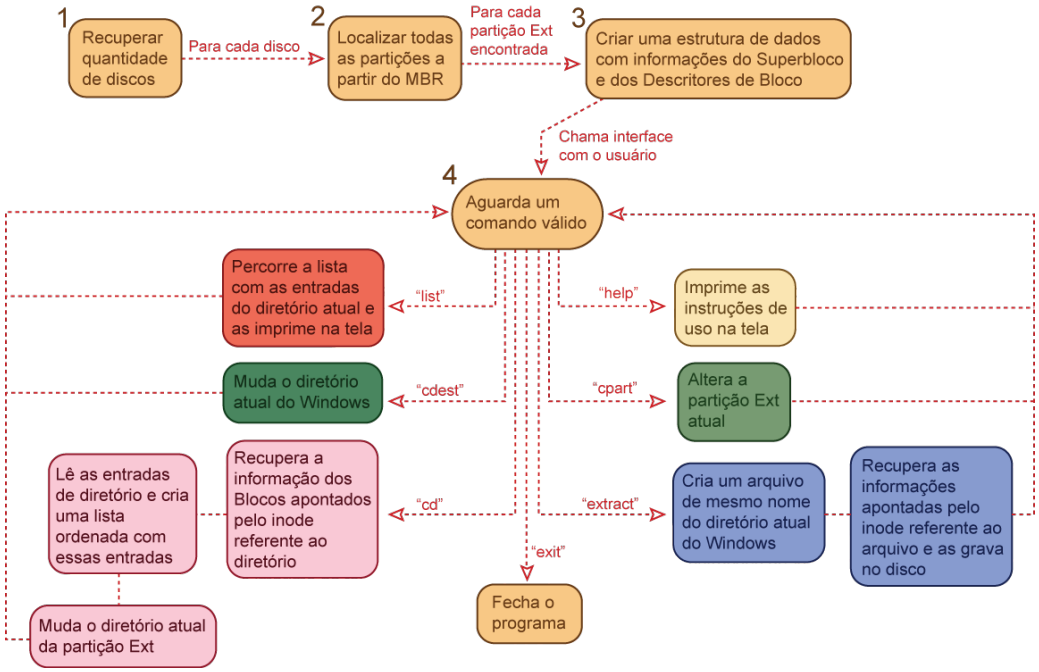

Figura 7: Diagrama mostrando o funcionamento do programa

4.0.1 Identificação de discos

O Windows 7 permite acesso aos discos físicos através dos arquivos da forma "`\\.\PhysicalDriveX`", onde X é um número natural. Ele atribui valores crescentes a X, ou seja, só existe um disco atrelado ao "`\\.\PhysicalDrive(n)`" se já tiver um disco atrelado ao arquivo "`\\.\PhysicalDrive(n-1)`".

Para identificarmos quantos e quais discos existem, basta tentarmos abrir esses arquivos; começando em 0 e incrementando o valor de X a cada tentativa. Na primeira falha teremos a quantidade de discos presentes e, consequentemente, como acessá-los.

4.0.2 Identificação das partições Ext

Toda partição Ext possui 0x83 como assinatura na sua entrada da tabela de partição, mas isso não significa que todas as partições com essa assinatura são do tipo Ext. Existem outros sistemas de arquivos que compartilham dessa mesma assinatura como, por exemplo, o ReiserFS e o Xiafs. Durante esta etapa todas as partições são identificadas e recebem uma estrutura padrão, com o seu tipo, posição e a qual disco pertence, mas somente aquelas cuja assinatura é 0x83 e a assinatura do superbloco é 0xEF53 (que se refere à partições Ext) ganham as estruturas mais complexas. A detecção foi feita dessa forma para permitir a expansão para partições de outros tipos.

A busca começa na tabela de partição do MBR, onde estão localizadas as primeiras 4 entradas. Precisamos ler cada uma delas e identificar, através da assinatura, se essa entrada é vazia (0x00) ou se é um ponteiro para um EBR (0x5 ou 0xF), ela não é colocada no vetor de partições. Entradas vazias são ignoradas. No caso de ponteiros para um EBR, a lista ligada é seguida e as partições adicionadas ao vetor até o final da lista ligada.

4.0.3 Recuperação das informações das partições Ext

Após todas as partições do disco serem encontradas, uma estrutura com as informações do superbloco necessárias para a extração dos arquivos é criada para cada partição Ext.

O processo descrito em toda a Etapa 2 e até este ponto da Etapa 3 é repetido para todos os discos encontrados na Etapa 1.

O superbloco nos dá informações valiosas, mas elas não são suficientes para conseguirmos extrair arquivos do sistema de arquivos. Ainda é impossível saber onde a tabela de inodes de cada Grupo de Bloco está localizada. Para isso, precisamos das informações dos Descritores de Grupo de Blocos, mas não faz sentido recuperar essas informações até decidirmos qual será a partição com a qual iremos trabalhar. Logo após a partição ser escolhida, um vetor contendo apontadores para o inicio da tabela de inodes de cada um dos Descritores é associado a esta partição. As informações do Grupo 0 estão na posição 0, as do Grupo 1 na posição 1 e assim sucessivamente.

O último passo dessa etapa consiste em preparar a pasta raiz para que o usuário, a partir dela, consiga navegar por todas as outras pastas e extrair os arquivos que ele desejar. Para tanto, é preciso copiar todos os blocos apontados pelo inode 2, que se refere à pasta raiz, criar uma estrutura para armazenar as entradas de diretório e copiar todas as entradas de diretório para essas estrutura. Depois disso as entradas são ordenadas e exibidas na tela.

4.0.4 Interação com o usuário

Na última etapa do programa, o usuário tem a opção de escolher uma série de comandos, para interagir com a partição Ext. Uma breve explicação sobre esses comandos, assim como funcionamento deles será apresentado logo mais:

- help – Exibe uma lista com o funcionamento dos comandos, de forma similar ao help dos programas padrão do Linux.
  Uso: help

- exit – Sai do programa.
  Uso: exit

- list – Lista todos os arquivos de um diretório. É semelhante ao programa “ls” com o parâmetro ‘-a’ do Linux. Para imprimir essa lista na tela, basta percorrermos a lista ordenada com todas as entradas de diretório e imprimirmos os atributos desejados de cada elemento.
  Uso: list

- cdest – Altera o diretório de destino, ou seja, o diretório do Windows onde os arquivos serão extraídos. A requisição de mudança é recebida e tratada pelo próprio sistema operacional, através da função SetCurrentDirectory().
  Uso: cdest “pasta de destino”

- cpart – Altera a partição atual. Retorna à Etapa 3.
  Uso: “cpart –n <número da partição>”

- cd – Altera o diretório atual da partição Ext. Ao contrário do comando “cd” do Linux e do Windows, esse comando só aceita um diretório como entrada e não um caminho até o diretório. Procuramos pelo nome ou número do diretório na estrutura que contém todas as entradas de diretório e nos certificamos de que ele é mesmo um diretório. Após encontrarmos, recuperamos a localização do inode que aponta para os blocos com as entradas de diretório. O resto do processo é similar ao final da Etapa 3, só que, nesse caso, temos um diretório qualquer e não mais a raiz.
  Uso: “cd <nome_do_diretório>” ou “cd –n <número do diretório>”.

- extract – Extraí o arquivo escolhido da partição Ext para a pasta atual do Windows. Primeiro realizamos um processo de busca ao nome do arquivo, para nos certificarmos de que ele está lá e não é um diretório. Depois disso pedimos para o Windows criar um arquivo com o mesmo nome que o arquivo que nós encontramos. Procuramos pelo inode desse diretório e começamos a copiar, byte a byte, o conteúdo dos blocos apontados pelo inode para o arquivo criado, até o ele atingir o tamanho certo.
  Uso: extract –n <numero_do_arquivo>

5 Testes

Durante a etapa de testes foram usados 4 notebooks com as seguintes características:

1. Notebook 1
    - Intel core i3 com Windows 7 (32 bits)
    - Partições: NTFS, NTFS, Ext4, Swap, NTFS
2. Notebook 2
    - Intel core i3 com Windows 7 (32 bits)
    - Partições: Ext4, NTFS, Ext4, Swap
3. Notebook 3
    - Intel core i3 com Windows 7 (32 bits)
    - Partições: NTFS, Swap, Ext4
4. Notebook 4
    - AMD Athlon 2 com Windows 7 (32 bits)
    - Partições: NTFS, NTFS, Ext4, Swap, Ext4

Para garantir a correção do algoritmo de extração, foram usados diversos arquivos com altíssima intolerância a erros, ou seja, arquivos que usam alguma forma de compactação que impede que ele seja interpretado corretamente em caso de falhas. Exemplos mais conhecidos desse tipo de arquivo são os ".pdf" e os ".rar". Nos testes foram usados arquivos de tamanhos variados, desde 1KB até algumas centenas de MB.

A seguir serão apresentadas algumas capturas de tela de um dos testes:

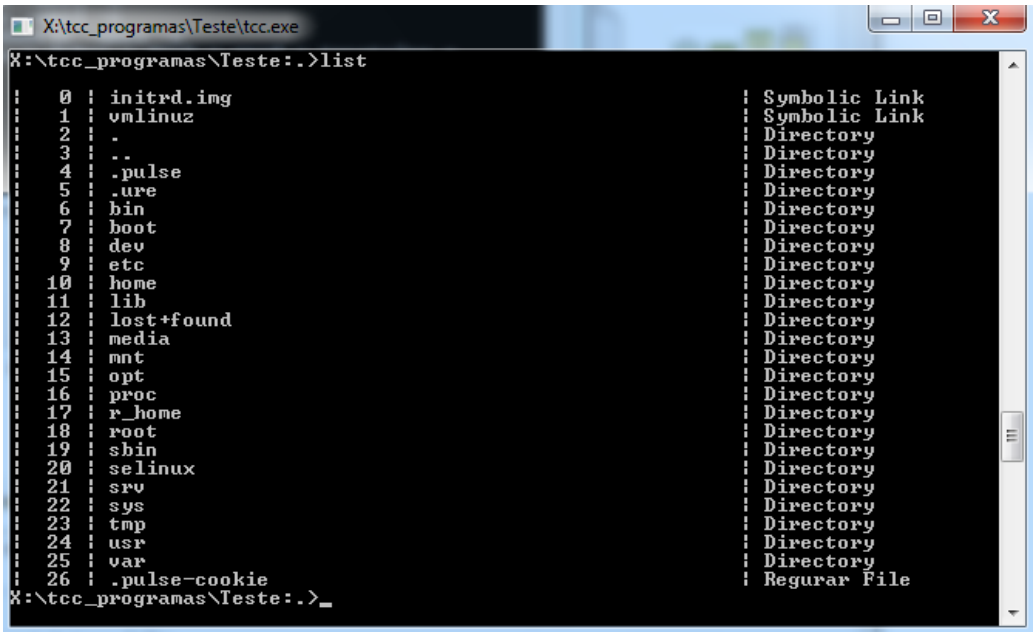

Figura 8: Execução do comando 'list'

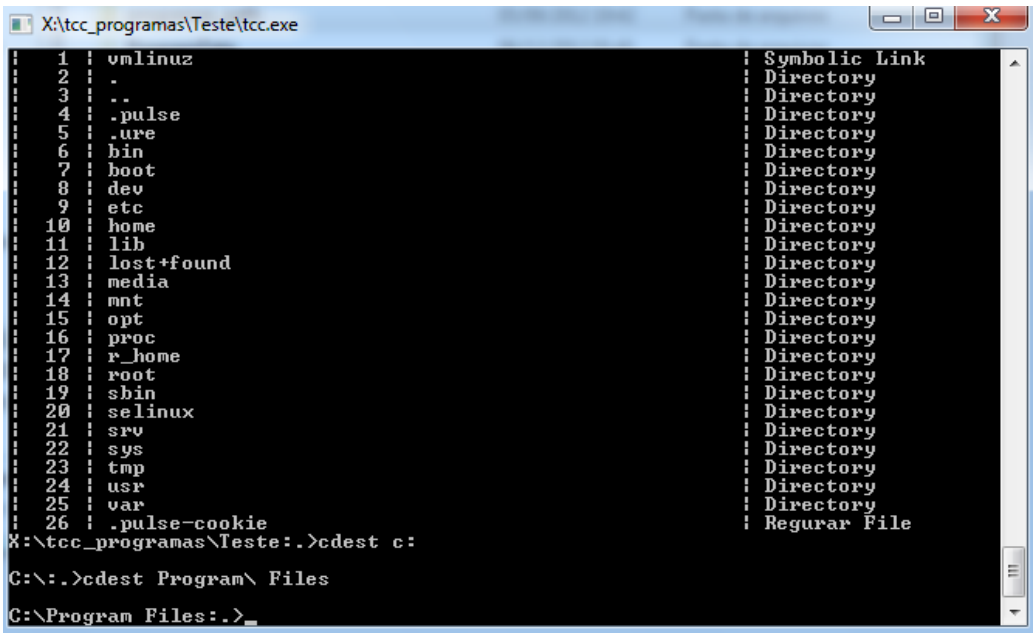

Figura 9: Alterando o diretório destino com o comando ’cdest’

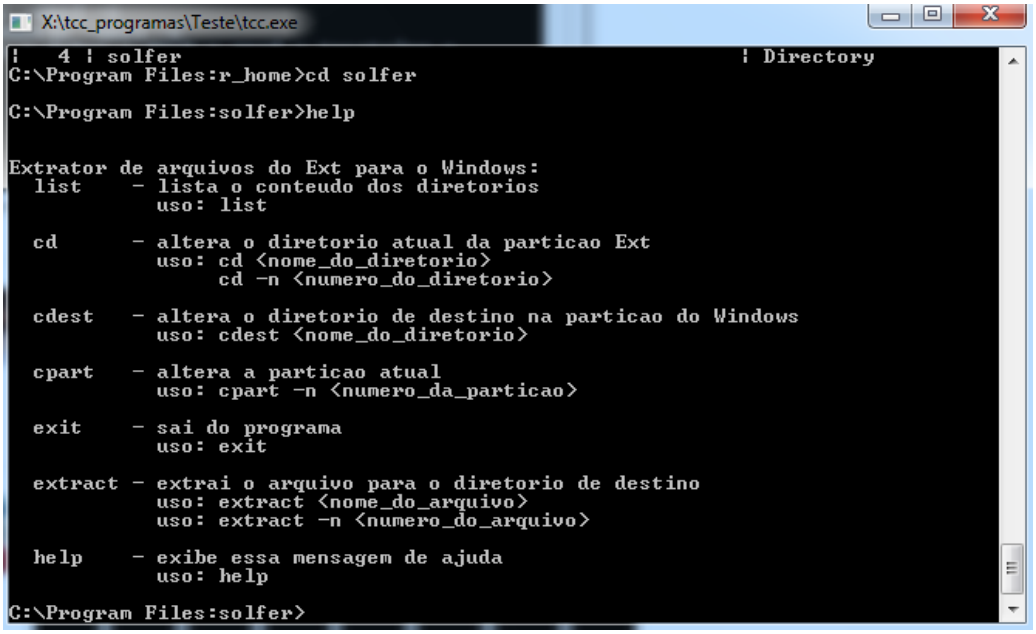

Figura 10: Exibindo a tela de ajuda com o comando ’help’

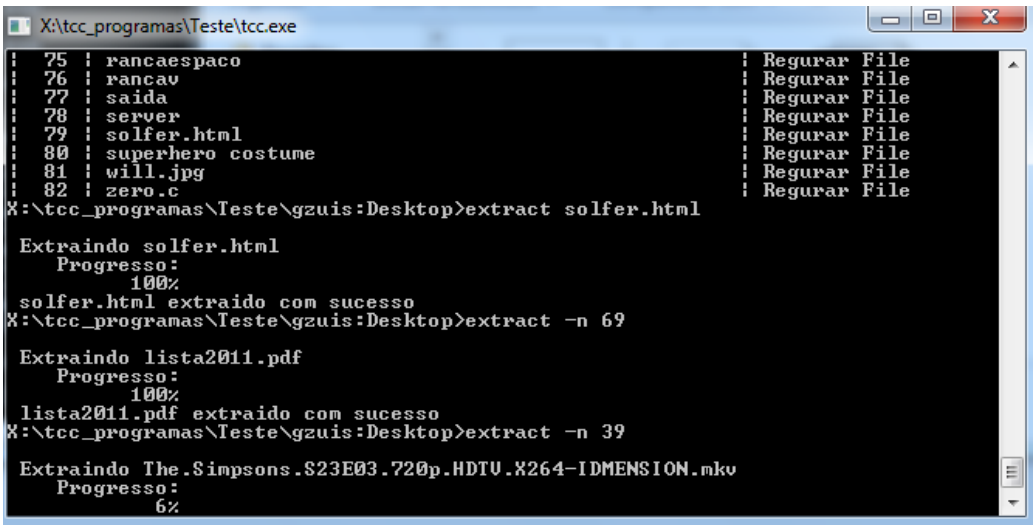

Figura 11: Extraindo um arquivo de mídia

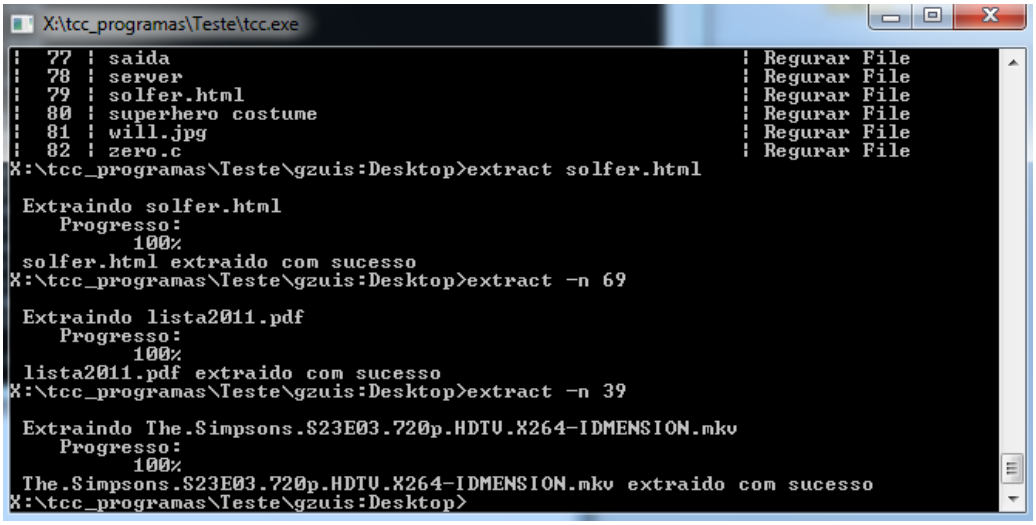

Figura 12: Mídia extraída com sucesso

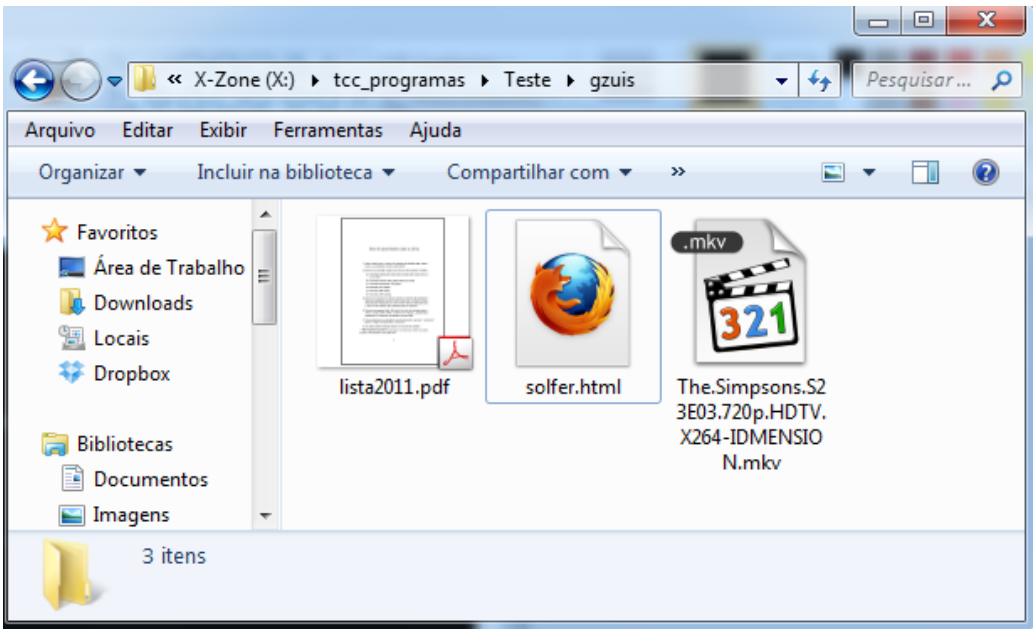

Figura 13: Arquivos extraídos sendo vistos pelo Windows Explorer

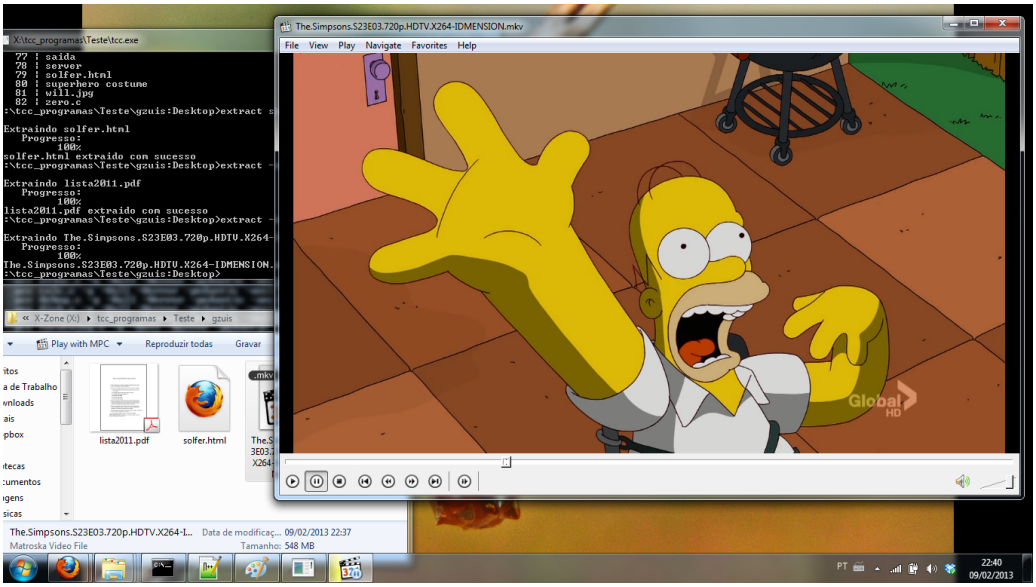

Figura 14: Mídia sendo reproduzida, sem falhas

6 Conclusão

Após estudar a fundo esses sistemas de arquivos e suas estruturas, posso afirmar que o Ext4 é realmente superior aos seus antecessores. Ver a evolução dos mecanismos, fruto de mais de 20 anos de trabalho e pesquisa, me deixou intrigado quanto a possibilidade de um Ext5. Quais estruturas seriam reaproveitadas? Como seriam os armazenados os dados? Será que conseguirão reduzir ainda mais a fragmentação?

Apesar de não ter conseguido cumprir com a minha proposta inicial, considero o resultado deste trabalho acadêmico satisfatório. Espero que ele possa servir como uma introdução às pessoas que, assim como eu, se interessam pelo funcionamento dos sistemas de arquivos.

7 Parte subjetiva

7.1 Desafios e frustrações

Durante o período de desenvolvimento do trabalho encontrei enumeras pedras pelo caminho. A maioria delas foram no âmbito acadêmico, mas as mais duras foram na vida pessoal.

No início, meus planos eram desenvolver um driver para Windows 7 com leitura e escrita. Um driver tem muito mais utilidade do que um extrator, já que os sistemas de arquivos Ext seriam vistos pelo sistema operacional. Passei três semanas das férias de julho alternando meu tempo entre tentar entender o funcionamento de drivers no Windows 7 e resolver uns problemas da IMEJr, empresa júnior do IME da qual eu sou diretor presidente.

Quando percebi que não conseguiria implementar um driver em tempo hábil, as férias já haviam terminado. Foi um pouco depois disso que conversei com o Gubi, meu orientador, e consegui mudar o tema.

Após a primeira pedra contornada, surgiu a pior de todas: o término do relacionamento amoroso. Esse evento me desestabilizou de todas as maneiras possíveis, mas prefiro não entrar em detalhes.

O processo de implementação, por incrível que pareça, não foi a parte mais difícil. É claro que foi bastante trabalhoso e consumiu muitas horas de pesquisa, desenvolvimento e depuração, mas foi bem divertido.

A ideia de manter um blog foi uma coisa que eu gostei bastante, no princípio. Até costumava brincar com os meus amigos dizendo que se eu não me formasse no IME, viraria blogueiro. Mas depois da mudança de servidor, o que era uma diversão, virou uma frustração. No começo eu até conseguia atualizar o blog, mesmo com a lentidão do servidor, mas existiam momentos em que o site simplesmente não mostrava o botão de nova postagem.

Citarei apenas mais um desafio: a língua portuguesa. Apesar de ser uma matéria obrigatória no IME, os alunos, no geral, não costumam manter uma boa relação com ela e eu não sou uma exceção. Para me preparar para a escrita da monografia, eu tive que retomar o gosto pela leitura. Sei que a minha escrita ainda é ruim, mas acredite, ela era muito pior.

7.2 Disciplinas cursadas

- **MAC110 – Introdução à Computação, MAC122 – Princípios de Desenvolvimento de Algoritmos**
  Sem as disciplinas introdutórias eu não teria a base necessária para programar nem o básico do TCC.

- **MACC 323 - Estrutura de Dados**
  A disciplina de Estrutura de Dados foi uma disciplina que me fez amadurecer muito como programador. Ela também introduz o uso de estruturas de dados mais complexas, como árvores e grafos.

- **MAC 336 - Criptografia para Segurança de Dados**
  Cursar criptografia junto com ED foi uma das melhores coisas que eu fiz na minha graduação. Essa matéria me fez evoluir muito como programador e me fez tomar gosto pela área de Segurança da Informação, que eu pretendo me dedicar profissionalmente em breve.

- **MAC 211 - Laboratório de Programação I**
  Em LabProg I, os alunos do curso têm a experiência de fazer o seu primeiro grande projeto, normalmente em C. Um dos objetivos da disciplina é mostrar ao aluno que é possível construir grandes projetos usando o paradigma de programação estrutural.

- **MAC 422 - Sistemas Operacionais**
  Foi a disciplina mais importante para o meu TCC. Nela eu aprendi alguns conceitos básicos de Sistemas de Arquivos e programação usando system calls. Além disso, foi graças ao último EP dessa disciplina (um programa que manipulava imagens FAT12, FAT16 e FAT32) que eu obtive a inspiração necessária para o tema do TCC e o gosto excessivo por escovação de bits.

- **MAC 328 - Algoritmos em Grafos**
  Apesar de ter usado apenas árvores e listas ligadas no meu TCC, essa disciplina me auxiliou muito no uso de ponteiros e habilidades com a linguagem C.

- **MAC 412 - Organização de Computadores**
  É a única matéria obrigatória de hardware que temos na graduação. Ela fornece ao aluno uma familiarização com o uso de flags, operações lógicas e números binários, muito usados no TCC. Além disso, a matéria está intimamente ligada com Sistemas Operacionais.

7.3 Comentários

Em primeiro lugar, gostaria de expressar o meu profundo descontentamento com o sistema operacional Windows. Como usuário eu até gosto dele, mas programar para essa plataforma sem usar as ferramentas da Microsoft é um pesadelo. A única parte que eu considerei realmente produtiva no meu breve contato com o mundo do desenvolvimento para Windows foi durante a minha fase de pesquisa sobre funcionamento de drivers. Eles podem rodar em modo Kernel e uma vez instalados, suas ações não são questionadas. Com isso dá pra fazer muita tela ficar girando, muita placa mãe apitar e por aí vai.

A necessidade em melhorar minha escrita fez com que eu retomasse o gosto pela leitura. No começo ler era uma tarefa forçada e os capítulos eram intermináveis, mas logo se tornou uma fonte de entretenimento. Isso fez com que eu recuperasse a curiosidade e a vontade de aprender coisas novas. Só por isso o TCC já valeu.

Durante o período das férias de começo de ano, em que eu vim ao IME quase todos os dias, percebi uma coisa: Não importa o quão triste eu esteja, quão desesperador um problema seja: Quando passo pelos portões da USP, eu me sinto feliz.

Cabe aqui um agradecimento a duas pessoas muito especiais: Gubi e William, vocês são, sem sombra de dúvida, as pessoas mais inteligentes que eu já conheci. Obrigado por me ajudarem e servirem de inspiração na minha vida.

Para concluir, gostaria de deixar a minha citação favorita como reflexão ao leitor:

> "Un ministre est excusable du mal qu'il fait, lorsque le gouvernail de l'État est forcé dans sa main par les tempêtes; mais dans le calme il est coupable de tout le bien qu'il ne fait pas." - Voltaire

Ou, em tradução livre:

> "Um governante pode ser perdoado por todo mal que ele causar, quando o leme do Estado for forçado por tempestades; Mas na calmaria, ele é culpado por todo o bem que ele não faz."

7.4 Próximos passos

- Adicionar suporte a novos Sistemas de Arquivos.
- Criar um sistema de recuperação de arquivos removidos.
- Adicionar escrita ao Sistema de Arquivos.
- Refatorar e remodularizar o código.
- Adicionar verificações de integridade dos arquivos e estruturas (CRC32, MD5 e SHA1).
- Adicionar extração de diretórios e múltiplos arquivos.
- Usar o conhecimento adquirido para fazer um driver de leitura e escrita.
- Permitir navegação direta em subdiretórios (ou seja, fazer com que "cd pasta1/pasta2/" funcione).
- Fazer a tecla Tab autocompletar os comandos, pastas e diretórios.
- Melhorar o gerenciamento de memória.
- Adicionar mais conteúdo à monografia.

8 Apêndice

8.1 Entradas do Superbloco

Inicio	Tamanho	Descrição
0x00	4	Quantidade total de inodes
0x04	4	Quantidade total de blocos
0x08	4	Quantidade de blocos reservados
0x0C	4	Quantidade de blocos livre
0x10	4	Quantidade de inodes livre
0x14	4	Primeiro bloco de dados
0x18	4	Log2 do tamanho de um bloco, em KB
0x1C	4	Log2 do tamanho do cluster, em blocos (Se o recurso bigalloc estiver ativo, 0 caso contrário)
0x20	4	Blocos por grupo
0x24	4	Fragmentos por grupo (obsoleto)
0x28	4	Inodes por grupo
0x2C	4	Horário da montagem (Unix Timestamp)
0x30	4	Horário de escrita (Unix Timestamp)
0x34	2	Número de vezes que o sistema foi montado desde a última verificação.
0x36	2	Número de montagens além do recomendável para verificação
0x38	2	Assinatura do Ext (0xEF53)
0x3A	2	Estado do Sistema de Arquivos (ver Flags do Superbloco)
0x3C	2	Comportamento ao detectar erros (ver Flags do Superbloco)
0x3E	2	Porção menor do nível de revisão
0x40	4	Horário da última verificação (Unix Timestamp)
0x44	4	Tempo máximo entre verificações, em segundos
0x48	4	Sistema Operacional (ver Flags do Superbloco)
0x4C	4	Nível de revisão (ver Flags do Superbloco)
0x50	2	UID padrão para os blocos reservados
0x52	2	GID padrão para os blocos reservados
0x54	2	Primeiro inode não reservado
0x58	4	Tamanho do inode, em bytes
0x5A	2	Número do grupo de blocos que contém esse Superbloco
0x5C	4	Flags de recursos compatíveis. O Kernel ainda pode ler/escrever se não entender algum dos recursos ativos (ver Flags do Superbloco)
0x60	4	Flags de recursos incompatíveis. O Kernel não pode ler/escrever se não entender algum dos recursos ativos (ver Flags do Superbloco)
0x64	4	Flags de recursos compatíveis somente para leitura. O Kernel ainda pode ler se não entender algum dos recursos ativos (ver Flags do Superbloco)
0x68	16	128-bit UUID do volume

Inicio	Tamanho	Descrição
0x78	16	Rótulo do volume
0x88	64	Diretório onde o sistema de arquivos foi montado pela última vez
0xC8	4	Usado pelo algoritmo de compressão
0xCC	1	Quantidade de blocos usados na prealocação de arquivos
0xCE	1	Quantidade de blocos usados na prealocação de diretórios
0xD0	2	Quantidade de entradas reservadas na Tabela de Descritores de Grupos (para futuras expansões)
0xE0	4	Número do inode do registro (journal)
0xE4	4	Número de dispositivo do registro. Usado apenas se o recurso de registro externo estiver ativo
0xE8	4	Início da lista de inodes órfãos a serem apagados
0xEC	16	Semente de hash da HTREE
0xFC	1	Algoritmo padrão de hash usado nos hashes de diretórios (ver Flags do Superbloco)
0xFD	1	Tipo de backup do registro
0xFE	2	Tamanho dos descritores de grupos, em bytes. Usado se a flag de incompatibilidade com recurso de 64bits estiver setada
0x100	4	Opções padrão de montagem
0x104	4	Primeiro grupo de blocos metabloco. Se o recurso estiver ativo
0x108	4	Horário da criação do sistema de arquivos (Unix Timestamp)
0x10C	68	Cópia de segurança dos primeiros 68 bytes do inode do registro
0x150	4	32bits mais significativos da quantidade de blocos
0x154	4	32bits mais significativos da quantidade de blocos reservados
0x158	4	32bits mais significativos da quantidade de blocos livres
0x15C	2	Tamanho mínimo de um inode
0x15E	2	Quantidade de bytes que novos inodes devem reservar
0x160	4	Flags diversas
0x164	2	Quantidade de blocos lidos ou escritos antes de mudar para o próximo disco(RAID)
0x166	2	Quantidade de segundos de espera na prevenção multi-montagem (MMP)
0x168	8	Quantidade de blocos para os dados da MMP
0x170	4	Quantidade de blocos lidos ou escritos antes de retornar a esse disco(RAID)
0x174	1	log2 da quantidade de grupos em um grupo flexível
0x175	1	<desconhecido>
0x176	2	<desconhecido>
0x178	8	Quantidade de KB escritos no sistema de arquivos desde a sua criação
0x180	4	Número do inode do snapshot ativo

Inicio	Tamanho	Descrição
0x184	4	ID sequencial do snapshot ativo
0x188	8	Quanitdade de blocos reservados para uso futuro do snapshot ativo
0x190	4	Número do inode da cabeça da list de snapshots no disco
0x194	4	Quantidade de erros vistos
0x198	4	Horário da primeira ocorrência de um erro (Unix Timestamp)
0x19C	4	Inode envolvido no primeiro erro
0x1A0	8	Número do bloco envolvido no primeiro erro
0x1C8	32	Nome da função onde o erro ocorreu
0x1CC	4	Número da linha onde o erro ocorreu
0x1D0	4	Horário em que ocorreu o erro mais recente
0x1D4	4	Número da linha em que o erro mais recente ocorreu
0x1D8	8	Número do bloco envolvido no erro mais recente
0x1E0	32	Nome da função onde o erro mais recente ocorreu
0x200	64	Opções de montagem (ASCIIZ)
0x240	4	Número do inode do arquivo de quota do usuário
0x244	4	Número do inode do arquivo de quota do grupo
0x248	4	Blocos de overhead no sistema de arquivos
0x24C	432	Margem até o final do Superbloco
0x3FC	4	Checksum do Superbloco

8.1.1 Flags e Identificadores do Superbloco

Posição 0x3A: Estado do Sistema de Arquivos – simultâneas:

- ◇ 0x0001 Desmontado corretamente
- ◇ 0x0002 Erros detectados
- ◇ 0x0004 Órfãos sendo recuperados

Posição 0x3C: Comportamento ao detectar erros – exclusivas:

- ◇ 1 Continue
- ◇ 2 Remonte somente para leitura
- ◇ 3 Pânico

Posição 0x48: Sistema Operacional – exclusivas:

- ◇ 0 Linux
- ◇ 1 Hurd
- ◇ 2 Massix
- ◇ 3 FreeBSD
- ◇ 4 Lites

Posição 0x4C: Nível de revisão – exclusivas:

- ◇ 0 Formato original
- ◇ 1 Formato v2 com tamanho dinâmico de inodes

Posição 0x5C: Flags de recursos compatíveis – simultâneas:

- ◇ 0x1 Prealocação de diretório (COMPAT_DIR_PREALLOC)
- ◇ 0x2 “imagic inodes” (COMPAT_IMAGIC_INODES)
- ◇ 0x4 Possui registro (COMPAT_HAS_JOURNAL)
- ◇ 0x8 Suporta atributos estendidos (COMPAT_EXT_ATTR)
- ◇ 0x10 Possui blocos reservados na tabela de descritores de grupo, para futuras expansões (COMPAT_RESIZE_INODE)
- ◇ 0x20 Possui índices de diretórios (COMPAT_DIR_INDEX)
- ◇ 0x40 Grupo de blocos ociosos (COMPAT_LAZY_BG)
- ◇ 0x80 Excluir inode. Não utilizado (COMPAT_EXCLUDE_INODE)
- ◇ 0x100 Excluir mapa de bits. Não utilizado (COMPAT_EXCLUDE_INODE)

Posição 0x60: Flags de recursos incompatíveis – simultâneas:

- 0x1 Compressão (INCOMPAT_COMPRESSION).
- 0x2 Entradas de diretório guardam o tipo do arquivo (INCOMPAT_FILETYPE).
- 0x4 Sistema de Arquivos precisa de reparo (INCOMPAT_RECOVER).
- 0x8 Sistema de arquivos tem um dispositivo de registro a parte (INCOMPAT_JOURNAL_DEV).
- 0x1 Possui Grupos de blocos com metablocos. (INCOMPAT_META_BG).
- 0x40 Arquivos utilizam Extent Trees (INCOMPAT_EXTENTS).
- 0x80 Permite que o sistema de arquivo tenha até 2ĝ4 blocos (INCOMPAT_64BIT).
- 0x200 Grupo de blocos flexíveis (INCOMPAT_FLEX_BG).
- 0x2000 Nunca usado (INCOMPAT_BG_USE_META_CSUM).
- 0x4000 Diretórios grandes >2GB ou altura > 3 na htree (INCOMPAT_LARGEDIR).
- 0x8000 Dados no inode (INCOMPAT_INLINE_DATA).

Posição 0x64: Flags de recursos compatíveis somente para leitura – simultâneas:

- 0x1 Superblocos esparsos. (COMPAT_SPARSE_SUPER).
- 0x2 O sistema de arquivos guarda arquivos maiores de 2GB (COMPAT_LARGE_FILE).
- 0x4 Não usado (COMPAT_BTREE_DIR).
- 0x8 O tamanho dos arquivos é dado em blocos. (COMPAT_HUGE_FILE)
- 0x10 Descritores de grupos possuem checksum. (COMPAT_GDT_CSUM).
- 0x20 Indica que o limite de 32000 subdiretórios não se aplica mais.(COMPAT_DIR_NLINK).
- 0x40 Indica a existência de inodes grandes. (COMPAT_EXTRA_ISIZE).
- 0x80 O sistema de arquivos tem um snapshot (COMPAT_HAS_SNAPSHOT).
- 0x100 Quota (COMPAT_QUOTA).
- 0x200 O sistema de arquivos suporta "bigalloc". (COMPAT_BIGALLOC).

Posição 0xFC: Identificadores do algoritmo de hash – exclusiva:

- 0x0 Legacy.
- 0x1 Half MD4.
- 0x2 Tea.
- 0x3 Legacy, unsigned.
- 0x4 Half MD4, unsigned.
- 0x5 Tea, unsigned.

8.2 Descritores de grupo de blocos

Inicio	Tamanho	Descrição
0x0	4	32bits menos significativos do mapa de bits de blocos
0x4	4	32bits menos significativos do mapa de bits de inodes
0x8	4	32bits menos significativos da tabela de inodes
0xC	2	16bits menos significativos da quantidade de blocos livres
0xE	2	16bits menos significativos da quantidade de inodes livres
0x10	2	16bits menos significativos da quantidade de diretórios
0x12	2	Flags do grupo de blocos
0x14	4	32bits menos significativos do mapa de bits de exclusão de snapshot
0x18	2	16bits menos significativos do checksum do mapa de bits de blocos
0x1A	2	16bits menos significativos do checksum do mapa de bits de inodes
0x1C	2	16bits menos significativos da quantidade de inodes não usados
0x1E	2	Checksum do descritor de grupo
0x20	4	32bits mais significativos do mapa de bits de blocos
0x24	4	32bits mais significativos do mapa de bits de inodes
0x28	4	32bits mais significativos da tabela de inodes
0x2C	2	16bits mais significativos da quantidade de blocos livres
0x2E	2	16bits mais significativos da quantidade de diretórios
0x30	2	16bits mais significativos da quantidade de diretórios
0x32	2	16bits mais significativos da quantidade de inodes não usados
0x34	4	32bits mais significativos do mapa de bits de exclusão de snapshot
0x38	2	16bits mais significativos do checksum do mapa de bits de blocos
0x3A	2	16bits menos significativos do checksum do mapa de bits de inodes
0x3C	4	Margem até o final do descritor

Flags dos descritores de grupo de blocos: – simultâneas

- 0x1 Tabela e mapa de bits de inodes não inicializados (EXT4_BG_INODE_UNINIT).
- 0x2 Mapa de bits de blocos não inicializada (EXT4_BG_BLOCK_UNINIT).
- 0x4 Tabela de inodes está zerada (EXT4_BG_INODE_ZEROED).

8.3 Tabela de inode

Inicio	Tamanho	Descrição
0x00	2	Flags de modos do arquivo
0x02	2	16 bits menos significativos do UID
0x04	4	32 bits menos significativos do tamanho do arquivo (em bytes)
0x08	4	Horário do último acesso (Unix Timestamp)
0x0C	4	Horário da última mudança no inode (Unix Timestamp)
0x10	4	Horário da última modificação de dados (Unix Timestamp)
0x14	4	Horário de remoção (Unix Timestamp)
0x18	2	16 bits menos significativos do GID
0x1A	2	Quantidade de hard links
0x1C	4	32 bits menos significativos da quantidade de blocos
0x20	4	Flags de inode
0x24	4	<desconhecido>
0x28	60	Indexação de blocos - Usa *Extent Tree* ou Blocos Diretos e Indiretos
0x64	4	32 bits menos significativos da Versão do arquivo (para o NFS)
0x68	4	32 bits menos significativos do bloco de atributo estendido
0x6C	4	32 bits mais significativos do tamanho do arquivo (em bytes)
0x70	4	Endereço do fragmento (obsoleto)
0x74	2	16 bits mais significativos da quantidade de blocos
0x76	2	16 bits mais significativos do bloco de atributo estendido
0x78	2	16 bits mais significativos do UID
0x7A	2	16 bits mais significativos do GID
0x7C	2	16 bits menos significativos do checksum do inode
0x7E	2	Não utilizado
0x80	2	Tamanho extra do inode (tamanho - 128)
0x82	2	16 bits mais significativos do checksum do inode
0x84	4	Bits extras para o horário de mudança do inode (Unix Timestamp)
0x88	4	Bits extras para o horário de modificação do arquivo (Unix Timestamp)
0x8C	4	Bits extras para o horário de ultimo acesso ao arquivo (Unix Timestamp)
0x90	4	Horário de criação do arquivo (Unix Timestamp)
0x94	4	Bits extras para o horário de criação do arquivo (Unix Timestamp)
0x98	4	32 bits mais significativos da Versão do arquivo

Flags do modo do arquivo: – simultâneas

- 0x1 Outros podem executar (S_IXOTH)
- 0x2 Outros podem escrevevr (S_IWOTH)
- 0x4 Outros podem ler(S_IROTH)
- 0x8 Membros do grupo podem executar(S_IXGRP)
- 0x10 Membros do grupo podem escrever(S_IWGRP)
- 0x20 Membros do grupo podem ler (S_IRGRP)
- 0x40 Dono pode executar(S_IXUSR)
- 0x80 Dono pode escrever (S_IWUSR)
- 0x100 Dono pode ler(S_IRUSR)
- 0x200 Sitcky bit (S_ISVTX)
- 0x400 Configura GID(S_ISGID)
- 0x800 Configura (S_ISUID)

Flags do modo do arquivo: – exclusivas

- 0x1000 FIFO (S_IFIFO)
- 0x2000 Dispositivo de caractere(S_IFCHR)
- 0x4000 Diretório (S_IFDIR)
- 0x6000 Dispositivo de bloco(S_IFBLK)
- 0x8000 Arquivo comum (S_IFREG)
- 0xA000 Link simbólico (S_IFLNK)
- 0xC000 Socket (S_IFSOCK)

Flags do inode: - simultâneas

- 0x8 Todas as escritas no arquivo precisam ser sincronas (EXT4_INODE_SYNC)
- 0x10 Arquivo é imutável (EXT4_INODE_IMMUTABLE)
- 0x20 Arquivo só pode ser anexado (EXT4_INODE_APPEND)
- 0x40 O utilitário dump(1) não deve despejar o arquivo (EXT4_INODE_NODUMP)
- 0x80 Não atualizar o horário de acesso (EXT4_INODE_NOATIME)
- 0x1000 Diretório usa índices com hash (EXT4_INODE_INDEX)
- 0x2000 Diretório mágico AFS (EXT4_INODE_IMAGIC)
- 0x4000 Dados do arquivo devem sempre ser escritos através do registro (EXT4_INODE_JOURNAL_DATA)
- 0x8000 Cauda do arquivo não deve ser fundida (EXT4_INODE_NOTAIL)
- 0x10000 Todas as entradas de diretório devem ser escritas sincronamente (EXT4_INODE_DIRSYNC)
- 0x20000 Topo da hierarquia de diretório (EXT4_INODE_TOPDIR)
- 0x40000 Este é um arquivo grande (EXT4_INODE_HUGE_FILE)
- 0x80000 Inode usa *Extent Tree* (EXT4_INODE_EXTENTS)
- 0x200000 Inode usado para grandes atributos estendidos (EXT4_INODE_EA_INODE)
- 0x00400000 Este arquivo possuí blocos alocados depois do EOF (EXT4_INODE_EOFBLOCKS)
- 0x80000000 Reservado para a biblioteca do Ext4 (EXT4_INODE_RESERVED)

Referências

[ALL] CARD, Rémy; TS'O, Theodore Y.; TWEEDIE, Stephen C.;
Design and Implementation of the Second Extended Filesystem
http://e2fsprogs.sourceforge.net/ext2intro.html

[SCT1] TWEEDIE, Stephen C.;
Journaling the Linux ext2fs Filesystem
http://e2fsprogs.sourceforge.net/journal-design.pdf

[SCT2] TWEEDIE, Stephen C.;
EXT3, Journaling Filesystem
http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html

[TSO] TS'O, Theodore Y.;
Planned Extensions to the Linux Ext2/Ext3 Filesystem
http://e2fsprogs.sourceforge.net/extensions-ext23/

[OSD] OSDev Wiki,
Ext2,
http://wiki.osdev.org/Ext2

[KER1] Ext4 Wiki,
Introdution to Partition Tables ,
http://thestarman.pcministry.com/asm/mbr/PartTables.htm

[KER2] Ext4 Wiki,
Frequently Asked Questions,
https://ext4.wiki.kernel.org/index.php/Frequently_Asked_Questions

[SR] Starman's Realm Wiki,
Ext4 Disk Layout,
https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout

[SOM] TANENBAUM, Andrew S.
Sistemas Operacionais Modernos,
Prentice Hall, 3 ed., 2010.

[MS] Windows Dev Center - Desktop,
File Management Reference (Windows),
http://msdn.microsoft.com/en-us/library/windows/desktop/aa364233%28v=vs.85%29.aspx