Análise dos sistemas de arquivos FAT 16 e FAT 32

Beatriz P. Martis, Eduardo S. Lira, Ewerton A. Assis

28 de Junho de 2013

Resumo

O presente trabalho, que comporá nota parcial na disciplina de Sistemas Operacionais 2, ministrada pelo professor Eduardo S. de Albuquerque, no Instituto de Informática da Universidade Federal de Goiás, tem como objetivo apresentar os sitesmas de arquivos FAT 16 e FAT 32; sistemas de arquivo de relevância histórica, inicialmente apresentados no sistema operacional Microsoft MS-DOS. A análise se pautará em suas questões arquiteturais e em alguns detalhes de implementação.

1 Introdução

2 Características do sistema FAT

A tabela de arquivos FAT é situada no início do volume. São armazenadas duas tabelas para o caso de uma ser danificada, além disso as tabelas e o diretório *root* ou raíz devem ser armazenados num local fixo para que os arquivos necessários para a inicialização do sistema sejam localizados corretamente [1].

Originalmente o sistema FAT foi desenvolvido para máquinas com arquitetura IBM PC, por isso os bytes das entradas FAT são armazenados em "little endian", assim, ao utilizar uma máquina "big endian" é preciso traduzí-los ao serem movidos dados do disco ou para outro disco [3].

Cada volume tem seu próprio FAT, o qual contém informações de alocação para cada arquivo no volume através de listas ligadas de unidades de alocação, isto é, *clusters* formados por setores em potência de 2. Por meio disso, o sistema também pode indicar quais são as unidade livres para serem utilizadas por um arquivo sendo criado ou extendido [2].

Desconsiderando os *clusters* reservados, o FAT16 é limitado por 2^{16} ou 65.536 *clusters*, já o FAT32 por 2^{32} ou mais de 4 bilhões de *clusters* [1] (mais detalhes na Seção 5 - Particionamento de disco). O conceito básico do sistema FAT, como pode ser visto na Figura **??**, é que, para cada arquivo ou diretório, é alocado uma estrutura de dados, chamada entrada de diretório, que contém o nome do arquivo, tamanho, endereço de início e outros metadados; seu conteúdo é

guardado em um ou mais *clusters*, que podem ser encontrados através da tabela FAT, a qual identifica o próximo *cluster* e identifica o status de alocação dos *clusters* [2].

O FAT surgiu como uma boa solução para o gerenciamento de disco, principalmente porque a tabela era pequena o suficiente para caber na memória, permitindo um acesso aleatório rápido, mas se tornou inviável com o aumento do tamanho da tabela em discos fixos, o que exigia paginação. Outro problema é que a informação a respeito de espaço livre era espalhada entre os setores do FAT, resultando numa alta fragmentação: para *clusters* grandes, era desperdiçado, em média, metade de um *cluster* em cada arquivo [2].

3 Estrutura do volume e particionamento

Um sistema FAT geralmente é organizado da seguinte forma [4]: um setor de boot, no início do volume (veja a subseção 3.1); as tabelas de alocação FAT1 e FAT2, sendo esta última uma redundância da primeira (veja mais detalhes na seção 4); o diretório *root* ou raíz; e por fim outros diretórios e todos os arquivos representados. As áreas FAT1 e FAT2 geralmente são agrupadas arquiteturalmente sob o título de área FAT e as áreas do diretório *root* e dos demais diretórios e arquivos sob o título de área de dados [2].

Na versão FAT12/16 o diretório *root* ou raíz encontra-se no início da área de dados; enquanto na versão FAT32 o diretório raíz pode estar em qualquer parte da área de dados — tal situação é considerada rara, tendo como padrão o uso da atribuição da versão FAT12/16 [2].

O sistema de arquivos é alocado em *clusters*, unidades de dados do sistema de arquivo [4], que são um grupo consecutivo de setores — o número de setores geralmente é em base 2 (1, 2, 4, 8, 16, 32 ou 64) e o número ("endereço") do *cluster* deve ser representável em até 12 bits na FAT12, em até 16 bits na FAT16 e em até 32 bits na FAT32 (embora apenas 28 bits sejam realmente usados) [2]. Curiosamente, os números de *cluster* começam com o número 2: não existem os *clusters* 0 e 1; além destes estarem todos na área de dados. O tamanho de um *cluster* é determinado, geralmente, pelo tamanho do volume de armazenamento. Essas restrições limitam o tamanho máximo de um volume de armazenamento para até 4GB, já que o tamanho mínimo do *cluster* — e, portanto, do setor — é de 512 bytes e o tamanho máximo de um *cluster* é de 64 kilobytes [4] — embora em outra fonte o tamanho máximo de um *cluster* seja 32 kilobytes [2]. Cada *cluster* é identificado como "não-usado", "em uso", "*bad cluster*" ou "último *cluster* em um arquivo/diretório"; informações que estão disponíveis na tabela de alocação (mais detalhes na seção 4).

Como as áreas FAT e o setor de boot não usam a estrutura de *cluster*, determinar a localização do primeiro *cluster* não é uma tarefa fácil [2]. Na versão FAT12/16 o primeiro *cluster* é o primeiro setor do diretório raíz, que é alocado quando o sistema de arquivo é criado; na versão FAT32, o primeiro *cluster* é o primeiro setor da área de dados. Descobrir o endereço S de um setor a partir do número C de um *cluster*, ou vice-versa, pode ser obtido a partir dos seguintes

cálculos [2]:

$S = (C - 2) \times$ (número de setores por *cluster*) + (setor do *cluster* 2);

e

$C = ((S -$ setor do *cluster* 2)/(número de setores por *cluster*) + 2.

O diretório raíz contém entradas para cada arquivo e diretório na raíz; o que distingue o diretório raíz dos demais diretórios é sua posição no volume e seu tamanho fixo: 512 entradas para volumes em disco rígido e um tamanho varíavel para disquetes, dependendo do tamanho do volume. Como será apresentado na seção 5 (página 5), diretórios são estruturas semelhantes aos arquivos, com estruturas próprias que os distinguem.

3.1 Setor de boot

O setor de boot, como em geral é encontrado em outros sistemas de arquivos, tem como finalidade armazenar código que será executado pela BIOS do sistema computacional durante a inicialização do disco. Localizado no primeiro setor do volume, além de conter código de execução, esta área do sistema de arquivos contém informações sobre o próprio sistema, como o tamanho das tabelas de alocação (áreas FAT) e demais informações sensíveis ao funcionamento do sistema [2]: o número de estruturas FAT e o tamanho destas estruturas; tamanho das estruturas de diretório; tamanho do *cluster*; dentre outras informações. Geralmente a versão é marcada por tag, em formato *string*, com os possíveis valores "FAT12", "FAT16", "FAT32" ou apenas "FAT". Fora esse marcador, não existe, dentre as informações do setor de boot, qualquer dado acurado para a versão do sistema de arquivos FAT; essa informação só pode ser obtida a partir de cálculos feitos sobre as informações armazenadas [2].

Na versão FAT12/16, o setor de boot ocupa um único setor, enquanto na versão FAT32 muitos outros setores podem ser armazenados. Na versão FAT32 é possível encontrar informações sobre o próximo *cluster* disponível e a quantidade de *clusters* disponíveis (na estrutura de dados FSINFO); o endereço do setor com uma cópia de segurança (*backup*) do setor de boot, geralmente no setor 6, que pode ser usado por ferramentas quando o setor original estiver corrompido; e o endereço do setor de início do diretório raíz.

Outras informações não essencias estão também disponíveis no setor de boot: o nome OEM (*OEM name* [2]), usado para indicar a ferramenta que criou o sistema de arquivos; um número serial de 4 bytes que determina quando o sistema de arquivos foi criado; e, como já mencionado, uma tag para determinar qual a versão do sistema de arquivo, embora tal informação não seja objetivamente correta. Outra informação importante, a qual é também armazenada no diretório raíz, é o rótulo do volume do sistema de arquivos e um rótulo do sistema de arquivo.

O código de execução, anteriormente mencionado, são instruções de *JUMP* que levam a máquina a executar os códigos de execução, armazenados nos bytes

62 a 509 nas versões FAT12 e FAT16 e nos bytes 90 a 509 na versão FAT32. Sistemas FAT geralmente apresentam código de execução, mesmo que estes volumes não sejam inicializáveis (*bootable*). A MBR, geralmente no início do disco ou dispositivo de armazenamento e fora da responsabilidade do sistema de arquivo, é responsável por inicializar a execução do código presente no setor de boot.

4 Tabela de alocação

A tabela de alocação no sistema FAT tem uma relevância chave na arquitetura deste sistema de arquivo — como o próprio nome do sistema de arquivos indica (FAT é acrônimo para *File allocation table*, ou tabela de alocação de arquivos, em tradução livre) — ao controlar o uso dos *clusters*.

Como foi apresentado na seção 3 (página 2), as tabelas de alocação são armazenadas nas áreas de disco FAT1 e FAT2, as quais, como também supracitado, não usam a estrutura lógica dos *clusters*. A FAT2 é uma replicação da FAT1 e tem por finalidade atender certo nível de segurança ao indicar possíveis inconsistências que possam surgir no sistema de arquivo. O tamanho da área FAT é geralmente calculado pelo tamanho da estrutura FAT e a quantidade de *clusters* no sistema, informações armazenadas no setor de boot. O número exato de FATs é geralmente obtido a partir do setor de boot.

As tabelas de alocação armazenadas na área FAT têm por finalidade indicar o status dos *clusters* da área de dados e como os arquivos e directórios são armazenados na área de dados. Cada entrada na tabela representa um *cluster* na área de dados: quando determinado *cluster* está disponível, seu valor na tabela é marcado com o valor 0; quando o *cluster* é marcado como danificado, os valores $0xff7$, $0xfff7$ e $0x0ffffff7$ são marcados na tabela para as versões FAT12, FAT16 e FAT32, respectivamente. Qualquer outro valor na tabela de alocação indica que o *cluster* está alocado, podendo ser o endereço de outro *cluster* da "cadeia de *clusters*" ("*clusters chain*") ou ser apenas um marcador de "final de estrutura" (diretório ou arquivo; "*end of file*" ou simplesmente *EOF*).

A estrutura de armazenamento das tabelas de alocação FAT assemelha-se a um vetor unidimensional, com tamando igual e fixo para todas as entradas. O vetor tem como quantidade de células o número de *clusters* da área de dados. O tamanho de cada célula varia de acordo com a versão do sistema de arquivo, como supracitado: 12 bits para FAT12, 16 bits para FAT16 e 32 bits para FAT32.

Os *clusters* endereçáveis começam a partir do *cluster* 2. As entradas 0 e 1 armazenam, tipica e respectivamente, uma cópia do tipo do dispositivo e o "estado sujo" ("dirty status") do sistema de arquivo, este indicando se houve um desligamento inapropriado do sistema ou erros de superfície encontrados — estes valores podem não ser acurados já que não são essenciais.

Os tópicos apresentados nessa seção englobam conceitos apresentados em [2].

5 Estrutura de diretório

Em se tratando da estrutura dos diretórios FAT, deve-se analisar dois aspectos: entradas de diretórios curtas, associadas a nomes de arquivos curtos, e longas, associadas a nome de arquivos longos. Uma entrada de diretório é uma estrutura que é alocada para cada arquivo e diretório. Ela tem 32 bytes de tamanho, e contém os atributos do arquivo, tamanho, cluster inicial, datas e horários. Entradas de diretório podem existir em qualquer lugar na área de dados, pois elas são guadadas nos clusters alocados para um diretório. No sistema de arquivos FAT, um diretório é considerado um tipo especial de arquivo. Serão tratadas aqui, primeiramente, as entradas curtas, e posteriormente as longas.

O único diretório especial, que deve estar sempre presente, é o diretório raiz. No FAT16, o diretório raiz está em um lugar fixo no disco, imediatamente depois da última tabela de alocação de arquivos, e tem um número fixo de setores. Já no FAT32, o diretório raiz pode ter um tamanho variável e é uma "cadeia de clusters", assim como qualquer outro diretório. [1]

Quando um novo arquivo ou diretório é criado, uma entrada de diretório no diretório pai é alocada para ele. Como a entrada de diretório tem um tamanho fixo, pode-se imaginar que o conteúdo de um diretório é uma tabela de entrada de diretórios. Diferente do que é feito com os clusters, não é dado um número único às entradas de diretórios para identificá-las. Em vez disso, o único padrão para endereçar uma entrada de diretório é o uso do nome completo do arquivo, ou diretório, alocado a ela. Além disso, na criação do diretório, um cluster também é alocado, e é todo preenchido com zeros. O campo de tamanho do diretório não é usado, e deve sempre ser zero. O único método para determinar o tamanho do diretório é percorrer a lista estruturada pelo tabela de alocação de arquivos, partindo do diretório, até o marcador de final de arquivo ser encontrado.

Se o diretório em questão não for o diretório raiz, as duas primeiras entradas em um são para os diretórios ".", usado para endereçar o diretório atual, e "..", utilizado para endereçar o diretório pai. [2]

Diferente de outros diretórios, o diretório raiz não tem marcação de data (time stamp), e não contem "." e ".." como as duas primeiras entradas no diretório.

A figura 5 especifica a estrutura das entradas de um diretório FAT

Algumas observações sobre o primeiro byte (DIR_Name[0]) de uma entrada de diretório FAT:

- DIR_Name[0] = 0xE5 - diretório vazio (não existe um nome de arquivo ou diretório nesta entrada);
- DIR_Name[0] = 0x00 - diretório vazio (como em 0xE5) e todos os bytes DIR_Name[0] de todas as estradas após esta também estão setados como 0, ou seja, não há diretórios alocados depois deste;
- DIR_Name[0] = 0x05 - o verdadeiro valor para este byte é 0xE5. 0xE5

Name	Offset (byte)	Tamanho (bytes)	Descrição
DIR_Name	0	11	Nome curto.
DIR_Attr	11	1	Atributos do arquivo: ATTR_READ_ONLY 0x01 ATTR_HIDDEN 0x02 ATTR_SYSTEM 0x04 ATTR_VOLUME_ID 0x08 ATTR_DIRECTORY 0x10 ATTR_ARCHIVE 0x20 ATTR_LONG_NAME ATTR_READ_ONLY | ATTR_HIDDEN | ATTR_SYSTEM | ATTR_VOLUME_ID Os dois bits mais altos do atributo são reservados, e devem ser sempre setados como 0 quando um arquivo é criado e nunca serem modificados.
DIR_NTRes	12	1	Reservado para uso do Windows NT. Setado como 0 quando o arquivo é criado, e nunca modificado ou utilizado depois disso.
DIR_CtrTimeTenth	13	1	Contém a contagem de décimos de segundo da hora de criação do arquivo. A granularidade da parte de segundos de DIR_CtrTime é 2 segundos, então este campo está entre 0-199, inclusive.
DIR_CtrTime	14	2	Hora que o arquivo foi criado.
Dir_CtrDate	16	2	Data que o arquivo foi criado.
DIR_LstAccDate	18	2	Data do último acesso. Note que não há "horário do último acesso". Esta é a data da última leitura ou escrita. No caso de uma escrita, este campo deve conter o mesmo valor que DIR_WrtDate.
DIR_FstClusHI	20	2	Maior palavra do primeiro cluster desta entrada (sempre 0 para FAT16).
DIR_WrtTime	22	2	Hora da última escrita. Note que a criação do arquivo é considerada uma escrita.
DIR_WrtDate	24	2	Data da última escrita. Note que a criação do arquivo é considerada uma escrita.
DIR_FstClusLO	26	2	Menor palavra do primeiro cluster da entrada.
DIR_FileSize	28	4	Palavra de 32 bits (DWORD) contendo o tamanho do arquivo em bytes.

Figura 1: Estrutura da Entrada de Diretório (adaptada de [3] - página 23)

representa um caracter válido no Japão, então quando este deve ser escrito, 0x05 se torna seu alias.

Na prática, o campo DIR_Name é quebrado em duas partes: uma principal, de 8 caracteres, contendo o nome, e uma secundária, de 3 caracteres, contendo a extensão.

Curiosidades:

- DIR_Name[0] não pode ser os caracteres: 0x20, 0x22, 0x2A, 0x2B, 0x2C, 0x2E, 0x2F, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x5B, 0x5C, 0x5D e 0x7C. Além disso, não são aceitos caracteres minúsculos;
- Há um "." implícito entre a parte principal e a parte secundária do nome que não está presente em DIR_Name;
- Nos diretórios FAT, todos os nomes são únicos.

DIR_Attr especifica atributos do arquivo:
ATTR_READ_ONLY - arquivo somente leitura, qualquer tentativa de escrita irá falhar;
ATTR_HIDDEN - os pedidos de listagem "regulares" deste diretório não devem mostrar este arquivo;
ATTR_SYSTEM - arquivo do sistema operacional;
ATTR_VOLUME_ID - deve haver apenas um "arquivo" no volume que tem este atributo setado, e este arquivo deve estar no diretório raiz. O nome deste arquivo é na verdade o nome que se dá ao volume. DIR_FstClusHL e DIR_FstClusLO devem sempre ser 0 para este arquivo.
ATTR_ARCHIVE - indica para o sistema quais arquivos sofreram modificação desde o último backup do sistema.

A combinação de bits do atributo ATTR_LONG_NAME indica que o "arquivo" é, na verdade, parte da entrada de um nome longo de outro arquivo. Mais a frente neste texto isso será melhor explicado.

As entradas longas de diretório são definidas como "entradas curtas com um atributo especial". Como dito anteriormente, uma entrada longa é uma entrada regular na qual o atributo ATT_LONG_NAME tem o valor ATTR_READ_ONLY — ATTR_HIDDEN — ATTR_SYSTEM — ATTR_VOLUME_ID verdadeiro. Quando uma entrada de diretório desta forma é encontrada, ela é tratada de forma diferenciada pelo sistema: ela é vista como uma parte de um conjunto de entradas de diretório que são associadas a uma única entrada curta. [3]

Um conjunto de entradas longas está sempre associado a uma entrada curta, que eles sempre precedem imediatamente. Entradas longas são pareadas com entradas curtas por uma razão: apenas entradas curtas são visíveis a versões anteriores do MS-DOS/Windows. Uma entrada longa nunca existe sozinha, por regra. Se uma entrada longa é encontrada sem ser pareada com uma entrada curta, ela é chamada de "orfã".

A tabela 5.1 mostra um conjunto de n diretórios com entradas longas, associados a uma única entrada curta.

Entrada	**Saída**
n-ésima entrada longa	LAST_LONG_ENTRY (0X40) — N
... entradas longas adicionais	...
primeira entrada longa	1
entrada curta associada com as entradas precedentes	(não se aplica)

Tabela 5.1: Sequência de entradas de diretório longas

Primeiro, cada membro de um conjunto de entradas longas é unicamente numerada, e o último membro do conjunto contém uma flag indicando que ele é de fato o último membro.

6 Nomes de arquivos

7 Recuperação e segurança

Uma desvantagem do sistema de arquivos FAT16 e FAT32 é a questão de segurança. Diferente de outros sistemas amplamente utilizados no mercado, como NTFS e EXT. O que é feito para tentar minimizar esse fator, é a utilização de uma cópia de backup da tabela de alocação como "sistema de segurança" para corrompimentos. Entretanto, este procedimento é ineficiente, pois uma queda de energia durante uma operação que modifique os metadados, pode tornar a partição inacessível, ou corromper severamente diversos arquivos. [**?**]

A seguir será discutida a recuperação de arquivos, e a checagem de consistência no sistema de arquivos.

7.1 Recuperação de Arquivos

Apesar de existirem ferramentas para a recuperação de arquivos no sistema de arquivos FAT há muito tempo, a documentação sobre este tópico é bem escassa, e a teoria é pouca quando se procura "o que deveria ser feito".

Quando um arquivo é deletado, a entrada de diretório é marcada como inutilizada, e as entradas na tabela de alocação para o cluster são setadas para zero. Para recuperar o arquivo, é necessário saber onde o arquivo começa, e o tamanho dele.

Pode-se tentar recuperar os dados lendo-se os mesmos a partir do cluster inicial, já conhecido. Uma ferramenta de recuperação tem duas opções quando a questão é escolher entre os clusters seguintes a ler. Pode-se ler cegamente a quantidade de dados necessária para o tamanho do arquivo, e ignorar o status de alocação dos dados, ou pode-se ler somente dos clusters não alocados. A segunda opção vai ser melhor sucedida que a primeira um maior número de vezes, pois ela vai recuperar alguns arquivos fragmentados. [2]

Observe a Figura 7.1, que mostra seis clusters de um sistema de arquivo, em três cenários diferentes. O arquivo tem 7094 bytes de tamanho, e cada cluster tem 2048 bytes, então o arquivo ocupa quatro clusters. O cluster inicial é o 56.

No primeiro cenário, os quatro clusters estão alocados de forma consecutiva. Neste caso, ambas opções irão recuperar corretamente os clusters de 56 a 59. No segundo caso, o arquivo foi fragmentado em três grupos, e os clusters entre os fragmentos, 57 e 60, estão alocados para outro arquivo na hora que o processo de recuperação é iniciado. Neste cenário, a opção 1 irá recuperar os clusters de 56 a 59, incorretamente incluindo o cluster 57. A opção 2 irá recuperar corretamente os clusters 56, 58, 59 e 61. Finalmente, no último caso, tem-se um cenário onde o arquivo está alocado nos mesmos clusters que anteriormente, mas os clusters entre os fragmentos não estão alocados quando o processo de recuperação é iniciado. Aqui, ambas opções irão incorretamente recuperar os clusters de 56 a 59.

Outros testes podem ser feitos, com diferentes cenários, e irá se observar que a opção 2, onde o status de alocação dos clusters é levado em consideração, pode

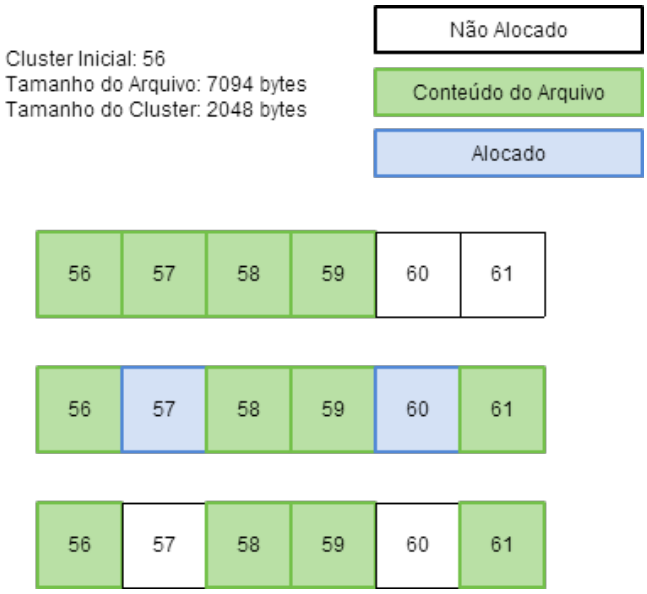

Figura 2: Exemplos de Cenários para Recuperação de Arquivos (adaptada de [2] pg. 182)

recuperar arquivos deletados em mais casos que a opção 1. [2] Como a alocação de clusters é feita por demanda, é muito provável que os diretórios estarão fragmentados no disco. Assim, diretórios são mais difíceis de se recuperar, a menos que ele seja o resultado de uma cópia, ou se o sistema de arquivos tiver sido desfragmentado recentemente.

Quando está se investigando um sistema de arquivos, é de grande valia realizar alguns testes de consistência para identificar arquivos de sistema corrompidos, ou dados que estão ocultos. No setor de boot, e em outras estruturas de dados na área reservada de um sistema de arquivos FAT, o teste de consistência deve verificar que alguns valores definidios estão em um intervalo apropriado, e também verificar os espaços não utilizados, procurando valores diferentes de zero. Por exemplo, existem vários setores na área reservada que não são usados em nenhum sistema de arquivos. Se um setor de boot de backup estiver disponível para o sistema de arquivos FAT32, uma checagem de consistência pode comparar os dois, e reportar qualquer diferença. A FAT de backup e a primária devem ser comparadas para verificar se elas possuem o mesmo valor. Cada entrada que é marcada como "bad" deve ser examinada pois a maioria dos discos conserta erros antes do sistema operacional notá-los.

O diretório raiz e seus subdiretórios devem ser examinados, e cada cadeia de clusters na FAT deve ser testada para se ter certeza que uma entrada de diretório alocada aponta para o início do um diretório. O teste reverso também deve ser feito para ter certeza que entradas de diretórios alocadas apontam para clusters alocados. Se múltiplas entradas de diretório apontam para uma cadeia

de clusters, é recomendado que todas sejam copiadas para uma nova localidade, e que os originais sejam deletados. O tamanho de uma cadeia de clusters deve ser o número de clusters necessários para o tamanho do arquivo.[2]

8 Conclusão

Referências

[1] Breecher, J. **The fat file system**. `http://web.cs.wpi.edu/~jb/CS502/samples/ExaminingTheFatFileSystem.ppt`, November 2008.

[2] Carrier, B. **File System Forensic Analysis**. Addison Wesley Professional, 2005.

[3] Microsoft, C. **Microsoft extensible firmware initiative: Fat32 file system specification**, December 2000.

[4] TechNet. **Chapter 17 - disk and file system basics**. `http://technet.microsoft.com/en-us/library/cc750198.aspx`, Junho 2013.