Undelete usando FAT32

Implementação de um programa undelete para sistemas de arquivos FAT32

Arthur do Prado Labaki - Marco Túlio Flores Melo - Vinnicius Pereira da Silva

Sumário

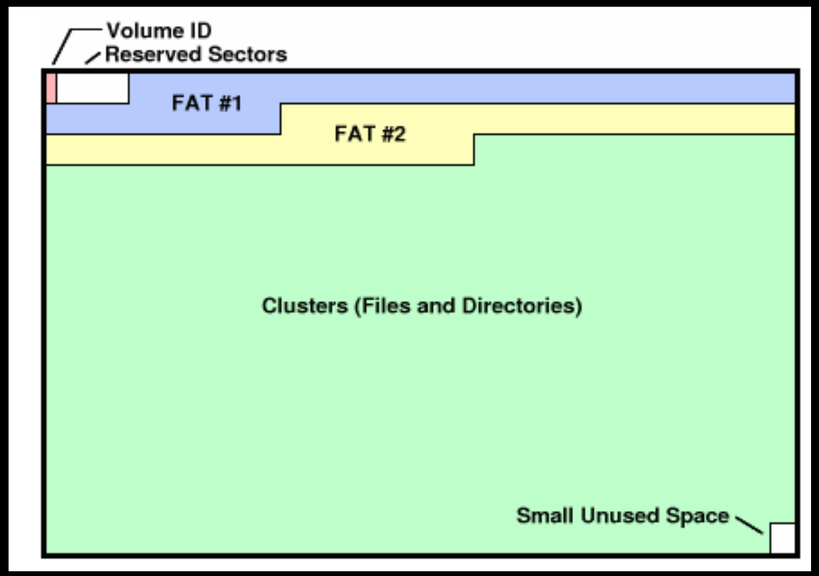
Introdução

- O FAT32 é um sistema de arquivos que gerencia o acesso em HDs e outras mídias. É um sistema que oferece bom desempenho mesmo em implementações leves, mas não oferece a mesma confiabilidade e desempenho de sistemas de arquivos modernos.
- A FAT32 tem um abrangente escopo de compatibilidade por quase todos os sistemas operacionais. Pois isso é muito usada hoje em dia.

FAT(File Allocation Table)

Estrutura Geral

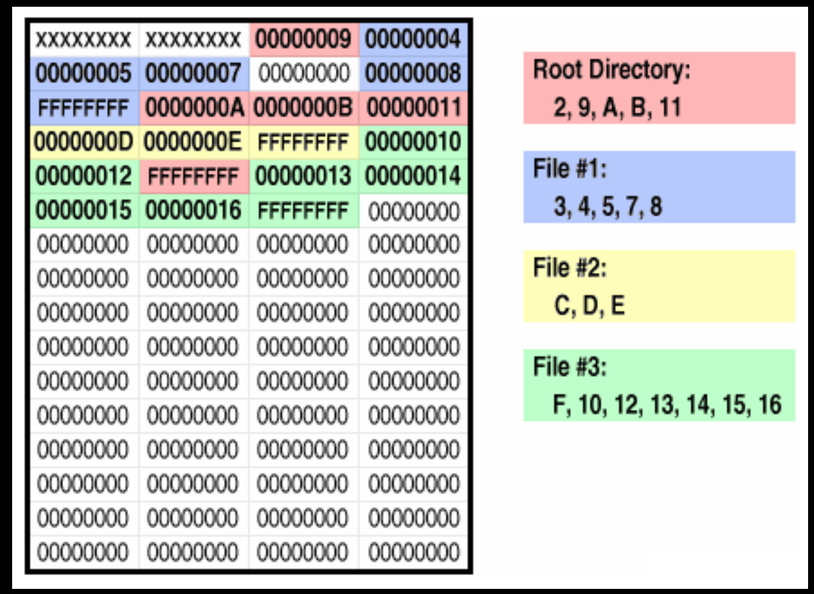
- O layout da FAT é bem simples, sendo o primeiro setor sempre o ID do Volume(Volume ID), que é seguido por um espaço não utilizado chamado de setores reservados(Reserved Sectors).
- O restante do arquivo é organizado em cluster, com um pequeno espaço não utilizado após o ultimo cluster.



FAT(File Allocation Table)

Estrutura Geral

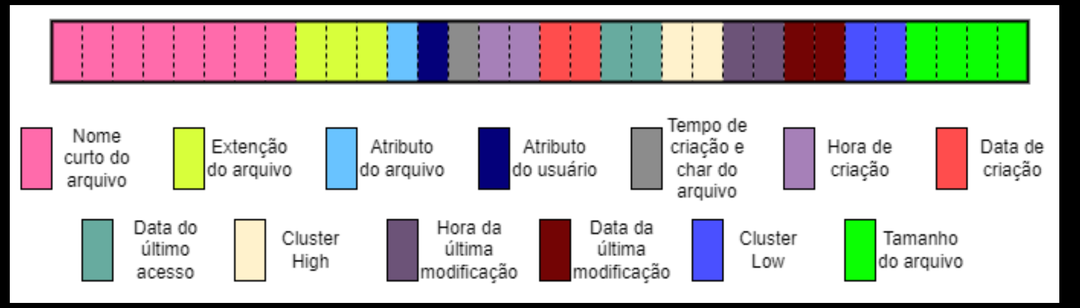
- Dos setores reservados, o primeiro é setor de inicialização (também chamado de Volume Boot Record ou simplesmente VBR). Ele inclui uma área chamada BIOS Parameter Block (BPB) que contém algumas informações do sistema de arquivos.
- Na região da FAT, normalmente contém duas cópias da Tabela de Alocação de Arquivos para verificação de redundância, embora raramente usada. Esses são mapas da Região de Dados, indicando quais clusters são usados por arquivos e diretórios
- Por fim, temos a região de dados, em que os dados reais do arquivo e do diretório são armazenados e ocupam a maior parte da partição.



FAT(File Allocation Table)

Tabela de diretórios

- O FAT32 normalmente inicia a tabela de diretório raiz no primeiro cluster da região de dados. Essa tabela de diretórios armazena informações sobre os arquivos e diretórios localizados no diretório raiz.
- Cada arquivo ou subdiretório armazenado dentro dele é representado por uma entrada de 32 bytes na tabela.
- Cada entrada registra o nome, extensão, atributos, o endereço do primeiro cluster dos dados do arquivo ou diretório, o tamanho do arquivo ou diretório, além de outras informações.



FAT(File Allocation Table)

Tabela de diretórios

- Algumas informações sobre nome e atributos são importantes para o entendimento dos códigos do programa proposto.
- É importante saber que o primeiro caractere do nome na estrutura de diretórios pode ser modificado. Se ele for 0xE5 representa um arquivo que já foi excluído e se ele for 0x00 ele representa que ele está disponível e nenhuma entrada subsequente está em uso.
- Também é importante saber o significado do conteúdo do atributo do arquivo, que está representado na tabela.

Bit	Máscara	Função
0	0x01	Somente leitura
1	0x02	Oculto
2	0x04	Sistema
3	0x08	ID do Volume
4	0x10	Subdiretório
5	0x20	Arquivo
6	0x40	Dispositivo
7	0x80	Reservado

Operações

Inserção

- A inserção de arquivos em FAT32 não é muito difícil de ser compreendida. Nela ocorre a inserção do arquivo na região de dados, alocando-os em um ou mais clusters em cadeias.
- É feito seu mapeamento na Tabela de Alocação de Arquivos, indicando a localização dos clusters usados por aquele arquivo.
- A escolha dos clusters não é feita de forma organizada, podendo fazer com que a cadeia de clusters pule alguns ou até voltando (começando no cluster 12 e seu próximo cluster ser o 7).
- Ainda é realizado a inserção de informações do arquivo na tabela de diretório raiz, armazenando em uma entrada de 32 bytes .

Operações
Remoção

- A remoção de arquivos não ocorre da forma com que a maioria das pessoas inexperientes acreditam. Na remoção, o arquivo não é apagado, ele apenas deixa de ser referenciado.
- Ele ainda é mantido, na região de dados, em seus respectivos clusters, além do mapa na tabela de alocação. Porém seu primeiro byte no registro de entrada na tabela de diretório raiz é alterado para 0xE5, sinalizando para o sistema que o arquivo acessado foi excluído.
- Posteriormente quando um novo arquivo for adicionado, ele poderá sobrescrever os dados daquele arquivo excluído, alterando alguns de seus clusters

Explicação e demonstração do código-fonte

```c
int getDeletedDirEntry(int fd, BootEntry* disk, char *filename, char *shaFile){ // tenta fazer o undelete
    unsigned int nEntries=0;          // numero de entradas de arquivos tabela de diretorio
    unsigned int currCluster = disk->BPB_RootClus; // Cluster onde o diretório raiz pode ser encontrado
    unsigned int totalPossibleEntry = (disk->BPB_SecPerClus * disk->BPB_BytsPerSec)/sizeof(DirEntry); //

    struct stat fs;                                 // struct para arquivos
    if(fstat(fd, &fs) == -1)                        // erro para ler a struct
    {
        perror("Erro ao ler o stat");
    }
    unsigned char* file_content  = mmap(NULL , fs.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0); //
    // cria uma area compartilhada, alterações feitas na região de mapeamento serão gravadas de volta no
    //alem de permitir o acesso de leitura e gravação
    int fileCount=0;    // numero de arquivos parecidos (distinguem do primeiro nome)
    int nEntries1=0;
    int currCluster1=0;
    DirEntry* dirEntry1;
```

```c
do{
    DirEntry* dirEntry = getclusterPtr(file_content,disk,currCluster);  // coloca na struct o cluster
    for(unsigned int m=0;m<totalPossibleEntry;m++){ //verifica todos arquivos no cluster
        if (dirEntry->DIR_Attr == 0x00){        // não tem mais diretorios para entrar (0x00 indica a partir dali n
            break;
        }
        if(isDirectory(dirEntry)==0 && dirEntry->DIR_Name[0] == 0xe5){     // não for diretorio e for excluido
            if (dirEntry->DIR_Name[1]==filename[1]){            //segundo caractere igual
                char *recFilename = getfilename(dirEntry);      // pega o nome do arquivo exluido
                if(strcmp(recFilename+1,filename+1)==0){        // compara os nomes, com exeção do primeiro caracte
                    if (shaFile){                // Necessario o uso do SHA
                        bool shaMatched = checkSHA(getShaOfFileContent(disk, dirEntry, file_content), shaFile);
                        if (shaMatched){         // sha é igual
                            fileCount=1;         // contagem de arquivos
                            nEntries1=nEntries;  // numero de entradas
                            dirEntry1=dirEntry;  //struct do arquivo
                            currCluster1=currCluster; // cluster do arquivo
                        }
                    }
                    else{                        // Não foi fornecido o sha1
                        if (fileCount<1){   // não tem nenhum arquivo semelhante ainda
                            nEntries1=nEntries;
                            dirEntry1=dirEntry;
                            currCluster1=currCluster;
                            fileCount++;    // aumenta em +1, pois pode existir outros
                        }
                        else{ // erro, mais de 1 arquivo parecido (Hello, Mello, Tello)
                            printf("%s: Erro de ambiguidade. Múltiplos candidatos encontrados.\n",filename);
                            fflush(stdout);
                            return 1;
                        }
                    }
                }
            }
        }
        dirEntry++;                         //proxima entrada do for
        nEntries+=sizeof(DirEntry);
    }
    unsigned int *fat = (unsigned int*)(file_content + disk->BPB_RsvdSecCnt*disk->BPB_BytsPerSec + 4*currCluster);
    //Tamanho em setores da área reservada*bytes por setor + 4*cluster atual -> calcula o proximo cluster na FAT
    if(*fat >= 0x0ffffff8 || *fat==0x00){   // não tem mais cluster para serem lidos ou parte não utilizada
        break;
    }
    currCluster=*fat;                       // proximo cluster
} while(1);
```

```
if (fileCount==1){                          // somente um arquivo encontrado
    updateRootDir(file_content, disk, filename, nEntries1); //recupera seu primeiro carac

    int n_Clusters = nOfContiguousCluster(disk, dirEntry1); // conta o numero de clusters
    if (n_Clusters>1){                      // mais do que um cluster para atualizar
        int startCluster = dirEntry1->DIR_FstClusHI << 16 | dirEntry1->DIR_FstClusLO;   /
        for(int i=1;i<n_Clusters;i++){      // vai colocar no mapa da fat os devidos prox
            updateFat(file_content , disk, startCluster, startCluster+1);
            startCluster++;
        }
        updateFat(file_content , disk, startCluster, 0x0ffffff8);   // coloca que ultimo
    }
    else                                    // um unico cluster
        updateFat(file_content , disk, currCluster1+1, 0x0ffffff8); // coloca que ultimo
    unmapDisk(file_content, fs.st_size);    // desmapeia o disco, para evitar alterações

    if (shaFile)                            // mostra na tela caso ocorra um sucesso
        printf("%s: Undelete com sucesso usando SHA-1\n",filename);
    else
        printf("%s: Undelete com sucesso\n",filename);
    fflush(stdout);
    return 1;
}
return -1;
```

Fim da apresentação

Perguntas?