|  |  |
| --- | --- |
| w | **FIPP – SISTEMAS DE INFORMAÇÃO – PRESIDENTE PRUDENTE** |

**ANDRESSA HISAE TSUKASAKI – R.A. 261742078  
GABRIEL MIGUEL NAVAS – R.A. 261741888  
GUSTAVO DILLIO LEITÃO – R.A. 261741748**

**ATIVIDADE DE PESQUISA SOBRE EXT3 X NTFS**

Presidente Prudente - SP

2019

**1.** ***Descreva a estrutura (struct real) de alocação de arquivos do NTFS, conhecida como MFT (Master File Table) e a*** ***estrutura do registro da entrada de diretório do NTFS. Quero a struct em linguagem C.***

**Alocação de arquivos:**  
typedef struct \_ATTRIBUTE\_RECORD\_HEADER {

ATTRIBUTE\_TYPE\_CODE TypeCode;

ULONG RecordLength;

UCHAR FormCode;

UCHAR NameLength;

USHORT NameOffset;

USHORT Flags;

USHORT Instance;

union {

struct {

ULONG ValueLength;

USHORT ValueOffset;

UCHAR Reserved[2];

} Resident;

struct {

VCN LowestVcn;

VCN HighestVcn;

USHORT MappingPairsOffset;

UCHAR Reserved[6];

LONGLONG AllocatedLength;

LONGLONG FileSize;

LONGLONG ValidDataLength;

LONGLONG TotalAllocated;

} Nonresident;

} Form;

} ATTRIBUTE\_RECORD\_HEADER, \*PATTRIBUTE\_RECORD\_HEADER;

**Fonte:** <https://docs.microsoft.com/en-us/windows/desktop/devnotes/attribute-record-header>

**Entrada de diretório:**

*typedef struct \_FILE\_NAME {*

*FILE\_REFERENCE ParentDirectory;*

*UCHAR Reserved[0x38];*

*UCHAR FileNameLength;*

*UCHAR Flags;*

*WCHAR FileName[1];*

*} FILE\_NAME, \*PFILE\_NAME;*

**Fonte:** <https://docs.microsoft.com/en-us/windows/desktop/devnotes/file-name>

**2.** ***Descreva a*** ***estrutura (struct real) de alocação do i-node e a*** ***estrutura do registro da entrada de diretório do Linux. Quero a struct em linguagem C.***

**Estrutura de alocação:**

struct inode {

[umode\_t](https://elixir.bootlin.com/linux/latest/ident/umode_t) i\_mode;

unsigned short i\_opflags;

[kuid\_t](https://elixir.bootlin.com/linux/latest/ident/kuid_t) i\_uid;

[kgid\_t](https://elixir.bootlin.com/linux/latest/ident/kgid_t) i\_gid;

unsigned int i\_flags;

#ifdef CONFIG\_FS\_POSIX\_ACL

struct [posix\_acl](https://elixir.bootlin.com/linux/latest/ident/posix_acl) \*i\_acl;

struct [posix\_acl](https://elixir.bootlin.com/linux/latest/ident/posix_acl) \*i\_default\_acl;

#endif

const struct [inode\_operations](https://elixir.bootlin.com/linux/latest/ident/inode_operations) \*i\_op;

struct [super\_block](https://elixir.bootlin.com/linux/latest/ident/super_block) \*i\_sb;

struct [address\_space](https://elixir.bootlin.com/linux/latest/ident/address_space) \*i\_mapping;

#ifdef CONFIG\_SECURITY

void \*i\_security;

#endif

/\* Stat data, not accessed from path walking \*/

unsigned long i\_ino;

/\*

\* Filesystems may only read i\_nlink directly. They shall use the

\* following functions for modification:

\*

\* (set|clear|inc|drop)\_nlink

\* inode\_(inc|dec)\_link\_count

\*/

union {

const unsigned int i\_nlink;

unsigned int \_\_i\_nlink;

};

[dev\_t](https://elixir.bootlin.com/linux/latest/ident/dev_t) i\_rdev;

[loff\_t](https://elixir.bootlin.com/linux/latest/ident/loff_t) i\_size;

struct [timespec64](https://elixir.bootlin.com/linux/latest/ident/timespec64) i\_atime;

struct [timespec64](https://elixir.bootlin.com/linux/latest/ident/timespec64) i\_mtime;

struct [timespec64](https://elixir.bootlin.com/linux/latest/ident/timespec64) i\_ctime;

[spinlock\_t](https://elixir.bootlin.com/linux/latest/ident/spinlock_t) i\_lock; /\* i\_blocks, i\_bytes, maybe i\_size \*/

unsigned short i\_bytes;

u8 i\_blkbits;

u8 i\_write\_hint;

[blkcnt\_t](https://elixir.bootlin.com/linux/latest/ident/blkcnt_t) i\_blocks;

#ifdef [\_\_NEED\_I\_SIZE\_ORDERED](https://elixir.bootlin.com/linux/latest/ident/__NEED_I_SIZE_ORDERED)

[seqcount\_t](https://elixir.bootlin.com/linux/latest/ident/seqcount_t) i\_size\_seqcount;

#endif

/\* Misc \*/

unsigned long i\_state;

struct [rw\_semaphore](https://elixir.bootlin.com/linux/latest/ident/rw_semaphore) i\_rwsem;

unsigned long dirtied\_when; /\* jiffies of first dirtying \*/

unsigned long dirtied\_time\_when;

struct [hlist\_node](https://elixir.bootlin.com/linux/latest/ident/hlist_node) i\_hash;

struct [list\_head](https://elixir.bootlin.com/linux/latest/ident/list_head) i\_io\_list; /\* backing dev IO list \*/

#ifdef CONFIG\_CGROUP\_WRITEBACK

struct [bdi\_writeback](https://elixir.bootlin.com/linux/latest/ident/bdi_writeback) \*i\_wb; /\* the associated cgroup wb \*/

/\* foreign inode detection, see wbc\_detach\_inode() \*/

int i\_wb\_frn\_winner;

u16 i\_wb\_frn\_avg\_time;

u16 i\_wb\_frn\_history;

#endif

struct [list\_head](https://elixir.bootlin.com/linux/latest/ident/list_head) i\_lru; /\* inode LRU list \*/

struct [list\_head](https://elixir.bootlin.com/linux/latest/ident/list_head) i\_sb\_list;

struct [list\_head](https://elixir.bootlin.com/linux/latest/ident/list_head) i\_wb\_list; /\* backing dev writeback list \*/

union {

struct [hlist\_head](https://elixir.bootlin.com/linux/latest/ident/hlist_head) i\_dentry;

struct [rcu\_head](https://elixir.bootlin.com/linux/latest/ident/rcu_head) i\_rcu;

};

[atomic64\_t](https://elixir.bootlin.com/linux/latest/ident/atomic64_t) i\_version;

[atomic\_t](https://elixir.bootlin.com/linux/latest/ident/atomic_t) i\_count;

[atomic\_t](https://elixir.bootlin.com/linux/latest/ident/atomic_t) i\_dio\_count;

[atomic\_t](https://elixir.bootlin.com/linux/latest/ident/atomic_t) i\_writecount;

#ifdef CONFIG\_IMA

[atomic\_t](https://elixir.bootlin.com/linux/latest/ident/atomic_t) i\_readcount; /\* struct files open RO \*/

#endif

const struct [file\_operations](https://elixir.bootlin.com/linux/latest/ident/file_operations) \*i\_fop; /\* former ->i\_op->default\_file\_ops \*/

struct [file\_lock\_context](https://elixir.bootlin.com/linux/latest/ident/file_lock_context) \*i\_flctx;

struct [address\_space](https://elixir.bootlin.com/linux/latest/ident/address_space) [i\_data](https://elixir.bootlin.com/linux/latest/ident/i_data);

struct [list\_head](https://elixir.bootlin.com/linux/latest/ident/list_head) i\_devices;

union {

struct [pipe\_inode\_info](https://elixir.bootlin.com/linux/latest/ident/pipe_inode_info) \*i\_pipe;

struct [block\_device](https://elixir.bootlin.com/linux/latest/ident/block_device) \*i\_bdev;

struct [cdev](https://elixir.bootlin.com/linux/latest/ident/cdev) \*i\_cdev;

char \*i\_link;

unsigned i\_dir\_seq;

};

[\_\_u32](https://elixir.bootlin.com/linux/latest/ident/__u32) i\_generation;

#ifdef CONFIG\_FSNOTIFY

[\_\_u32](https://elixir.bootlin.com/linux/latest/ident/__u32) i\_fsnotify\_mask; /\* all events this inode cares about \*/

struct [fsnotify\_mark\_connector](https://elixir.bootlin.com/linux/latest/ident/fsnotify_mark_connector) [\_\_rcu](https://elixir.bootlin.com/linux/latest/ident/__rcu) \*i\_fsnotify\_marks;

#endif

#if [IS\_ENABLED](https://elixir.bootlin.com/linux/latest/ident/IS_ENABLED)(CONFIG\_FS\_ENCRYPTION)

struct [fscrypt\_info](https://elixir.bootlin.com/linux/latest/ident/fscrypt_info) \*i\_crypt\_info;

#endif

void \*i\_private; /\* fs or device private pointer \*/

};

**Fonte:** https://elixir.bootlin.com/linux/latest/source/include/linux/fs.h

**Estrutura do registro da entrada de diretório do** linux**:**

*#define* [*EXT4\_NAME\_LEN*](https://elixir.bootlin.com/linux/latest/ident/EXT4_NAME_LEN) *255*

*struct* [*ext4\_dir\_entry*](https://elixir.bootlin.com/linux/latest/ident/ext4_dir_entry) *{*

[*\_\_le32*](https://elixir.bootlin.com/linux/latest/ident/__le32) *inode; /\* Inode number \*/*

[*\_\_le16*](https://elixir.bootlin.com/linux/latest/ident/__le16) *rec\_len; /\* Directory entry length \*/*

[*\_\_le16*](https://elixir.bootlin.com/linux/latest/ident/__le16)[*name\_len*](https://elixir.bootlin.com/linux/latest/ident/name_len)*; /\* Name length \*/*

*char name[*[*EXT4\_NAME\_LEN*](https://elixir.bootlin.com/linux/latest/ident/EXT4_NAME_LEN)*]; /\* File name \*/*

*};*

**Fonte:** https://elixir.bootlin.com/linux/latest/source/fs/ext4/ext4.h

**3.** ***Compare um i-node a um registro da MFT em relação aos seguintes aspectos:*a) *Onde são armazenados os atributos em cada uma dessas estruturas de alocação?***

No caso do Windows, como cada registro da MFT (master file table) descreve somente um arquivo ou diretório, neste registro há os atributos com todos atributos.   
  
No caso do Linux, todos atributos de arquivo ficam dentro de um i-node no qual descreve todo um arquivo ou diretório.

**b) *Em sistemas baseados em i-nodes, quando um arquivo cresce além do limite da estrutura de alocação, são utilizados blocos indiretos para acomodar os endereços do arquivo. Explique o que precisa ser feito quando um arquivo cresce além do limite de espaço de um registro da MFT.***

Se um arquivo aumentar de tamanho, será necessário, algumas vezes, usar dois ou mais registros da MFT para brigar a lista de todos os blocos. Nesse caso, o primeiro registro MFT, chamado de registro-base, aponta para os outros registros da MFT. Um mapa de bits faz o acompanhamento de quais entradas da MFT estão livres.

**3. *Descreva o conceito de grupo de blocos do EXT3. Qual a vantagem do EXT3 utilizar grupos de blocos ao invés da organização tradicional dos sistemas de arquivos (metadados no início do disco e dados de arquivos e diretórios no restante do disco)?***

O disco formatado por ext3, é formado por blocos, onde o primeiro bloco (bloco 0) não é utilizado pelo Linux (contém código de inicializar o computador). Após o bloco 0, a partição do disco é dividida em grupos de blocos, desconsiderando onde vão cair os limites do cilindro.

Descrito em ordem, cada bloco contém:

* Superbloco: espaço responsável por guardar a informações sobre o layout do sistema de arquivos incluindo o número de i-nodes, o número de blocos de disco e o começo da lista de blocos de disco livres (geralmente algumas centenas de entradas).
* Descritor do grupo: contém informações sobre a localização dos mapas de bits, o número de blocos livres e i-nodes no grupo, e o número de diretórios no grupo.
* Mapa de bits e Mapa de bits do i-node: Mapas de bits são usados para controlar os blocos livres e i-nodos livres.
* I-nodes: Eles são numerados de 1 até algum máximo. Cada i-nodo tem 128 bytes de comprimento e descreve exatamente um arquivo. Um i-nodo contém informações de contabilidade, assim como informações suficientes para localizar todos os blocos de disco que contêm os dados do arquivo.
* Blocos de dados: Todos os arquivos e diretórios estão armazenados aqui. Se um arquivo ou diretório consiste em mais do que um bloco, os blocos não precisam ser contíguos no disco. Na realidade, os blocos de um arquivo grande têm uma maior chance de disseminar-se por todo o disco.

A Grande vantagem de se utilizar esse tipo de organização em grupo de blocos, é que quando os arquivos crescem, são colocados junto ao grupo de blocos, preferivelmente logo após o último bloco anterior. Além disso, quando um novo arquivo é inserido em um diretório, o ext3 tenta colocar dentro do mesmo grupo de blocos em que está seu diretório. Quando novos blocos de arquivos são alocados, ext3 também pré-aloca um número (oito) de blocos adicionais para aquele arquivo, de maneira a minimizar a fragmentação de arquivos devido a futuras operações de escrita. Esse esquema equilibra a carga do sistema de arquivos através do disco inteiro. Ele também tem um bom desempenho devido a suas tendências para colocação e fragmentação reduzida.

**Referências**:

* **Tanenbaum, Andrew S. Sistemas Operacionais Modernos**. 2ed. São Paulo: Prentice Hall, 2003.
* **Tanenbaum, Andrew S. Sistemas Operacionais Modernos** 3ed. São Paulo: Pearson Prentice Hall, 2009.
* **Estrutura de alocação do NTFS:** <https://docs.microsoft.com/en-us/windows/desktop/devnotes/attribute-record-header>
* **Estrutura do registro da entrada de diretório do NTFS:** <https://docs.microsoft.com/en-us/windows/desktop/devnotes/file-name>
* **Estrutura de alocação do Linux: Fonte:** https://elixir.bootlin.com/linux/latest/source/include/linux/fs.h
* **Estrutura do registro da entrada de diretório do Linux:** https://elixir.bootlin.com/linux/latest/source/fs/ext4/ext4.h