
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**João Pedro Soares da Franca – PE3021114**

**Estrutura de Dados 2**

# Sistema de Arquivos

Presidente Epitácio - SP

2024

# Sistema de Arquivos Estrutura de Dados 2

O desenvolvimento de um sistema de arquivos constitui uma tarefa essencial na concepção de sistemas operacionais, uma vez que é responsável pela organização, armazenamento e recuperação de dados em dispositivos de armazenamento. Um sistema de arquivos eficiente deve possibilitar o acesso e a manipulação de arquivos de forma ágil e segura, assegurando a integridade dos dados. No âmbito deste projeto, propõe-se a implementação de um sistema de arquivos utilizando a linguagem C, fundamentado em uma estrutura de dados projetada para gerenciar blocos de dados e i-nodes, com suporte à criação, leitura e manipulação de arquivos e diretórios. A seleção de uma estrutura de dados apropriada é determinante para garantir tanto a eficiência quanto a usabilidade do sistema.

A estrutura de dados concebida para este sistema de arquivos é composta por diversas entidades interligadas, incluindo blocos de dados, i-nodes e diretórios. Cada bloco de dados é representado pela estrutura sBlock (Figura 1), que compreende um endereço, um indicador de status (livre ou ocupado) e um vetor de caracteres destinado ao armazenamento dos dados do bloco.

```c
typedef struct sBlock {
    char address[70];
    int status; // 0 free | 1 busy
    char data[BLOCK_SIZE];
} Block;
```

***Figura 1.*** Representação da estrutura de blocos (sBlock).

Os i-nodes, representados pela estrutura sINode (Figura 2), armazenam metadados relativos aos arquivos e diretórios, tais como tipo, status, nome e tamanho. Cada i-node é capaz de referenciar até 15 blocos de dados, permitindo uma alocação eficiente de espaço para arquivos de tamanhos variados.

```c
typedef struct sINode {
    int id;
    char type; // 'r' regular file | 'd' directory file
    int status; // 0 free | 1 busy
    char name[MAX_FILENAME];
    long int size;
    int block_count;
    Block blocks[15];
} INode;
```

***Figura 2.*** Representação da estrutura de i-nodes (sINode).

Complementarmente, a estrutura sDirectory (Figura 3 e Figura 4) possibilita a organização hierárquica dos arquivos, armazenando informações sobre subdiretórios e a lista de i-nodes associados a cada diretório. Essa abordagem modular favorece a manipulação e a navegação dentro do sistema de arquivos.

```c
struct sDirectory;

typedef struct sDirectoryList {
    struct sDirectory *directory;
    struct sDirectoryList *next;
} DirectoryList;

typedef struct sDirectory {
    INode *inode;
    INodeList *iNodeList;
    char name[MAX_FILENAME];
    int childs_cont;
    struct sDirectory *parent; // ..
    struct sDirectoryList *childs; // sub directories
} Directory;
```

***Figura 3.*** Representação da estrutura de diretórios (sDirectory).

A decisão de utilizar uma combinação de blocos de dados e i-nodes no simulador do sistema de arquivos fundamenta-se em conceitos clássicos de sistemas

de arquivos, amplamente discutidos na disciplina de Sistemas Operacionais. Os i-nodes desempenham um papel crucial em diversos sistemas de arquivos, pois permitem a separação entre os metadados do arquivo e os dados efetivamente armazenados no disco. Essa separação promove a eficiência na gestão de informações como permissões de acesso e localização dos dados, eliminando a necessidade de redundâncias. Por outro lado, a estrutura baseada em blocos de dados viabiliza a alocação dinâmica de espaço, essencial para a eficiência do sistema ao lidar com arquivos de tamanhos variados.

Adicionalmente, a implementação de listas encadeadas para gerenciar blocos livres e i-nodes livres, representadas pelas estruturas sFreeBlock (Figura 4) e sFreeINode (Figura 5), oferece uma solução eficiente para a reutilização de recursos no sistema de arquivos. Quando um arquivo é removido, seu i-node e os blocos de dados associados são marcados como livres e reincorporados a essas listas, permitindo que novos arquivos sejam alocados sem a necessidade de buscas exaustivas por espaço disponível. Tal abordagem não apenas incrementa a eficiência do sistema, mas também simplifica a implementação de operações como criação, exclusão e renomeação de arquivos.

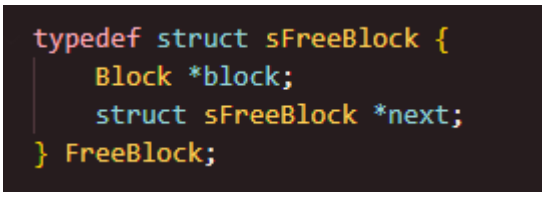


***Figura 4.*** Representação da estrutura de blocos livres (sFreeBlock).

```c
typedef struct sFreeINode {
    INode *inode;
    struct sFreeINode *next;
} FreeINode;
```

***Figura 5.*** Representação da estrutura de i-nodes livres (sFreeINode).

Em síntese, a estrutura de dados escolhida para este sistema de arquivos em C reflete uma análise criteriosa das demandas por eficiência, escalabilidade e organização dos dados. A integração de i-nodes e blocos de dados, aliada ao uso de listas encadeadas para o gerenciamento de recursos livres, constitui uma base sólida para a implementação de um sistema de arquivos robusto e funcional. Essa abordagem não apenas atende aos requisitos propostos no projeto, mas também está em conformidade com as melhores práticas e conceitos fundamentais abordados na disciplina de Sistemas Operacionais, garantindo a adequação do sistema às exigências de um ambiente de armazenamento contemporâneo.