**Especificação de SO – Neo File System**

O NFS (Neo File System) foi desenvolvido na disciplina de SO (Sistemas Operacionais, 2016) na UNIOESTE (Universidade Estadual do Oeste do Paraná), pelos alunos João Pedro Silveira, Bruno Luiz Casarotto e Letícia Torres.

1. [Decisões de projeto](#un32p9jmj22)
   1. [Organização da repartição](#lbbvhl6knifz)
   2. [Organização do HASH](#fwlkvr1fym92)
2. [Detalhes de implementação](#mj7xreu2k0zf)
   1. [INFO](#a035w15oee0d)
   2. [Entry](#huwl137cova6)
   3. [Diretórios](#gyiw4rwbxnuc)
3. [Guia de programação](#6ao9wy6itybw)

1. **Decisões de projeto**

Esta seção apresenta a organização do sistema de arquivos sem explicitar muitos detalhes técnicos de implementação.

* 1. **Organização da repartição**

Foi decidido dividir a partição do disco em grupos, e os grupos em blocos, sendo a quantidade de blocos e grupos uma potência de 2.

Com exceção do primeiro grupo, todos os grupos possuem a seguinte estrutura: blocos de dados, blocos reservados para o hashmap e mais blocos de dados. O HASH fica no meio do grupo para diminuir a movimentação do cabeçote.

O primeiro grupo é um caso especial, porque este contém a estrutura INFO no seu primeiro bloco. Portanto ele possui um bloco de dados a menos que os outros grupos.

O diretório root é alocado no bloco seguinte à INFO.

Exemplificação do disco é feita na Tabela 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Grupo 0 | | | | | Grupo 1 … N | | |
| INFO | Root | DATA | HASH | DATA | DATA | HASH | DATA |

Tabela 1 - Divisão dos grupos.

* 1. **Organização do HASH**

O HASH é um compilado de indexadores de blocos. Há três possíveis sentidos de uma valor do HASH:

1. Valor 0 indica que o bloco indexado por aquela posição está livre e pode ser ocupado com dados.
2. Valor diferente do número do bloco indexado indica que os dados da entrada continuam no bloco que o índice aponta.

Ex:

|  |  |  |
| --- | --- | --- |
| 2 | 3 | 3 |

O índice 1 tem valor 2, isso quer dizer que os dados da entrada continuam no bloco 2, o índice 2 tem valor 3, então tem mais dados da entrada no bloco 3.

1. Valor igual ao do número do bloco indexado indica que os dados da entrada terminam neste bloco.

Ex:

|  |  |  |
| --- | --- | --- |
| 2 | 3 | 3 |

O índice 1 tem valor 2, isso quer dizer que os dados da entrada continuam no bloco 2, o índice 2 tem valor 3, então tem mais dados da entrada no bloco 3. O índice 3 tem valor 3, isto quer dizer que não há mais blocos alocados para a entrada.

1. **Detalhes de implementação**

Essa seção é destinada a apresentar os detalhes de implementação do sistema, especificado na linguagem de programação C++.

* 1. **INFO**

A estrutura de dados da INFO é descrita como na Tabela 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Tamanho (bytes) | Nome do campo | Descrição |
| 0 | 1 | block\_size | Expoente de 2 para obter o tamanho de um bloco. |
| 1 | 1 | n\_blocks\_per\_group | Expoente de 2 para obter o número de blocos que cada grupo comporta |
| 2 | 2 | n\_groups | Número de grupos na repartição |

Tabela 2 - Estrutura de Dados INFO.

Os dados apresentados são minimalistas, porque block\_size e n\_blocks\_per\_group precisam ser calculados para obter-se o valor real deles, o que se resume em fazer o cálculo de 2 elevado ao valor do campo.

* 1. **Entry**

A estrutura de dados de Entry é descrita como na Tabela 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Tamanho (bytes) | Nome do campo | Descrição |
| 0 | 32 | name | Nome da entrada. |
| 32 | 1 | name\_length | Tamanho do nome da entrada. Negativo → entrada excluída. |
| 33 | 1 | attributes | Atributos da entrada. |
| 34 | 6 | creation\_time | Data e hora de criação da entrada. |
| 40 | 6 | modification\_time | Data e hora da última modificação feita na entrada. |
| 46 | 6 | last\_access\_time | Data e hora do último acesso na entrada. |
| 52 | 8 | size | Tamanho da entrada em bytes. |
| 60 | 4 | first\_block | Primeiro bloco do conteúdo da entrada. |

Tabela 3 - Estrutura de Dados Entry.

Os campos attributes, creation\_time, modification\_time e last\_access\_time são tipos complexos, sendo attributes uma enum e os campos time uma struct. A enum attributes segue a seguir na Tabela 4 e a Struct time na Tabela 5.

|  |  |  |
| --- | --- | --- |
| Valor | Nome do atributo | Descrição do atributo |
| 1 | READ\_ONLY | Indica que a entrada é somente leitura. |
| 2 | HIDDEN | Indica que a entrada está escondida. |
| 4 | DIRECTORY | Indica que a entrada é um diretório. |
| 8 | FILE | Indica que a entrada é um arquivo. |
| 16 | READ | Indica que é possível ler os dados da entrada. |
| 32 | WRITE | Indica que é possível escrever nos dados da entrada. |
| 64 | EXECUTABLE | Indica que é possível executar a entrada. |

Tabela 4 - Enum attributes.

|  |  |  |  |
| --- | --- | --- | --- |
| Offset (Bits) | Size (Bits) | Campo | Descrição |
| 0 | 12 | year | Ano. (0 - 4095) |
| 12 | 4 | month | Mês. |
| 16 | 5 | day | Dia. |
| 21 | 5 | hour | Hora. |
| 26 | 6 | minute | Minuto. |
| 32 | 6 | second | Segundo. |
| 38 | 10 | millisecond | Milisegundo. |

Tabela 5 - Struct date\_time.

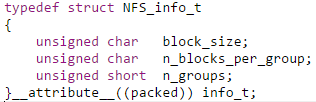
* 1. **Diretórios**

Os diretórios são organizados da seguinte maneira: a primeira entrada é o ‘.’, diretório corrente, a segunda entrada é o ‘..’, diretório pai, seguido por n entradas de arquivos e diretórios.

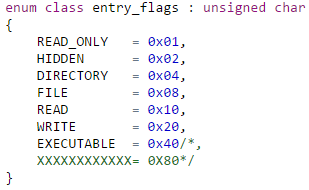
Para ler um subdiretório basta ler o(s) bloco(s) de dado(s) da entrada e listar o conteúdo dele.

1. **Guia de Programação**

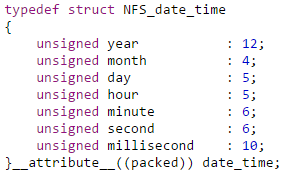
Struct para INFO:



Struct para os atributos de entrada:



Struct para date\_time:



Struct para entrada:

