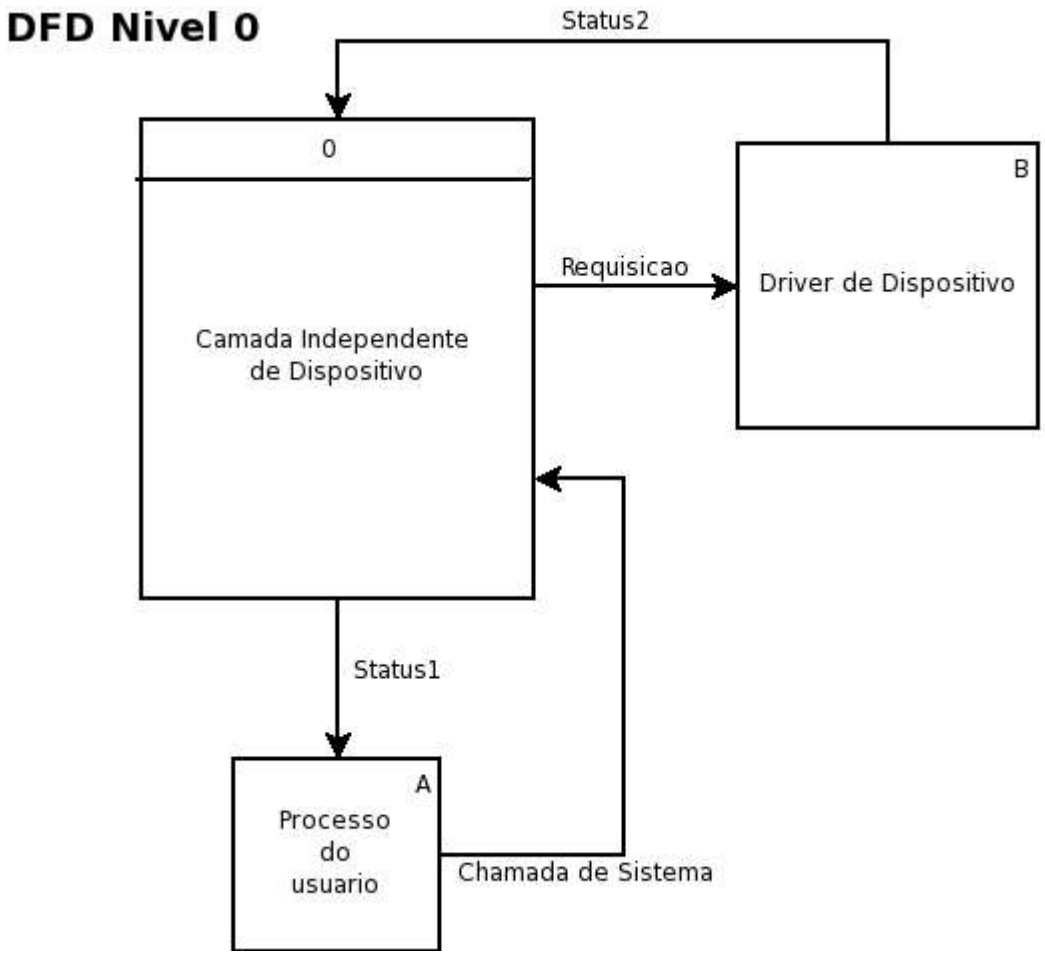
Projeto: Camada Independente de Dispositivo

Introdução

Esse documento tem como finalidade demonstrar como será implementada a Camada Independente de Software.

Estrutura

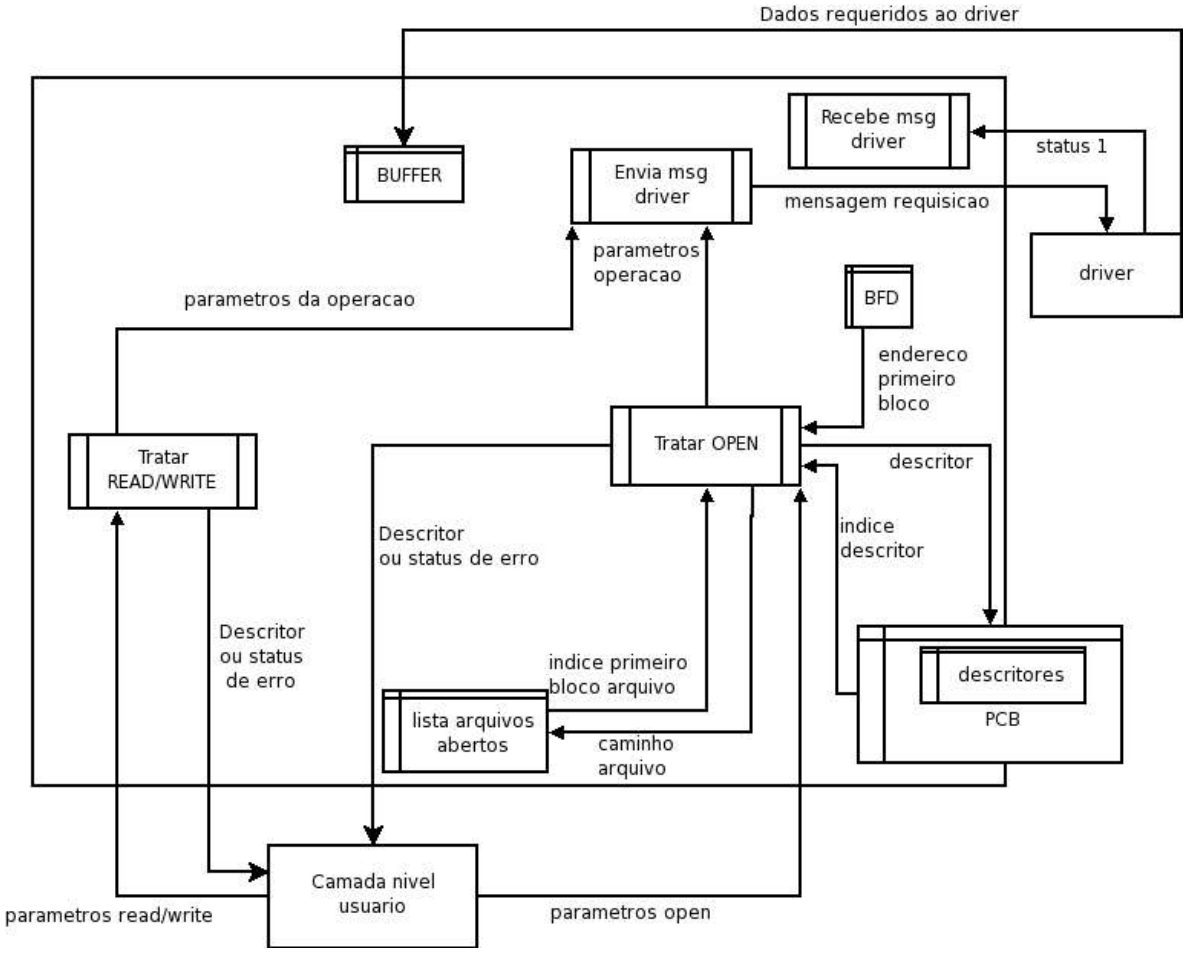
A camada independente de software é interligada da seguinte forma:



Camada Independente de Dispositivo: É a camada próxima do usuário ou operador, vide especificação. A camada tem como princípio enviar requisições para o driver de dispositivo, se assim necessário e receber um status de retorno ou o dado requerido.

Processo do usuário: Entidade que faz requisições a camada independente de dispositivo, requisições essas como: open, read, write.

Driver de dispositivo: É a camada de software responsável por requerer blocos de dados da controladora de disco, é a camada que "busca" os dados no dispositivo, pois conhece todas as especificações do mesmo. Essa camada se comunica diretamente com a controladora requerendo bloco para ser lido.



Consultar/Atualizar descritores e verificar cache: função que consulta e/ou atualiza os descritores do PCB correspondente do processo que fez a chamada de sistema. Analisa as chamadas de sistemas e chama a função Executa chamada de sistema passando os parametros necessários para cada chamada. Essa função consulta também se o dado requerido já existe na cache, não necessitando fazer requisição a função Executar chamada de sistema.

Executar chamada de sistema: função que faz requisição ao driver de dispositivo em relação a dados necessitados pela chamada de sistema feito pelo processo do usuário. Essa função faz chamada a função que trata a FAT, chama o driver e atualiza o cache de dados.

Tratar FAT: função que consulta a tabela de alocação de arquivos (FAT) do disco.

Buffer de dados: estrutura de dados que armazena dados lidos ou devem ser escritos do disco.
Lista de arquivos abertos: estrutura de dados que armazena uma lista de arquivos abertos no sistema operacional.

FAT: file allocation table, tabela de alocação de arquivos. Tabela existente em disco, nos seus primeiros blocos. Possui se os blocos do disco estão ocupados (com dados).

PCB: estrutura que armazena dados detalhados de um processo (vide especificação).

Lista de descritores: estrutura de dados que armazena descritores para os arquivos abertos pelo processo e que contém o bloco inicial, flag de leitura e escrita do arquivo.

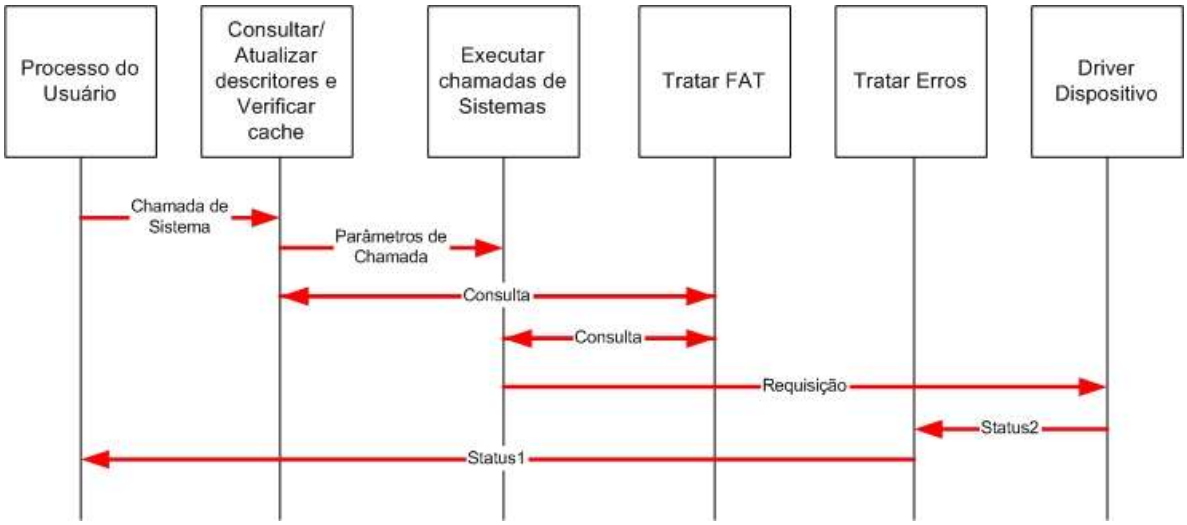
Comunicação entre as camadas

A comunicação entre a camada independente de dispositivo (CSID) e o processo do usuário é direta, ou seja, cada processo conterá uma cópia da CSID e irá fazer requisições diretas a ela. A CSID é um módulo que é incluido no processo do usuário.

A comunicação entre a CSID e o driver de dispositivo será feita através de uma fila de mensagem. Cada requisição estará numa mensagem, sendo que cada requisição conterá os seguintes parâmetros.

- Número do cluster – long inteiro
- Tipo de operação – inteiro // 0 quando for operação de leitura ou 1 quando for escrita
- Posição memória – long inteiro //onde os dados serão lidos no caso escrita ou onde os dados serão escritos no caso de uma leitura
- Pid – long inteiro // identificador do processo dono da requisição
- Número de bytes a serem escritos ou lidos - inteiro

Diagrama de controle



Estruturas de dados

Requisição

Número do Cluster	Tipo de Operação	Posição de Memória	PID	Número de Bytes

- Número do cluster – inteiro
- Tipo de operação – inteiro // 0 quando for operação de leitura ou 1 quando for escrita
- Posição memória – inteiro //onde os dados serão lidos no caso escrita ou onde os dados serão escritos no caso de uma leitura
- Pid – inteiro // identificador do processo dono da requisição
- Número de bytes a serem escritos ou lidos – inteiro

Lista de arquivos abertos

É uma estrtura de dados (struct) contendo:

- Primeiro bloco: Primeiro bloco em disco do arquivo aberto
- Contador: Número de vezes que aquele arquivo foi aberto no sistema

Descritor

Estrutura que contém:

- Inteiro handler: indeficação do arquivo
- Ponteiro para o buffer de dados do arquivo
- Posicação na lista de arquivos abertos
- Ponteiro para a lista de arquivos abertos

PCB

Estrutura que contém:

- Char caminho corrente: Diretório corrente daonde o processo está
- Lista de descritores dos arquivos abertos pelo processo

Requisição

Estrutura que armazenará os dados que serão requeridos junto ao driver e/ou serão úteis para a requisição no driver

- Operação: inteiro que indica a operação a ser feita pelo driver
- Index: inteiro que indica qual bloco será lido
- Id share bloco: numero do bloco compartilhado (buffer) que armazenará os dados lidos
- Pid do processo

BUFFER

Estrutura de memória compartilhada que armazenará os blocos lidos pelo driver. Contém os campos:

- Index: idenficação no bfd dessa entrada nesse bloco
- Tipo: tipo arquivo, diretório arquivo etc
- Nome: nome do arquivo ou diretório

Algorítmos dos módulos

Envia requisição driver

```
comentario: Envia requisicao ao driver atraves de uma fila de mensagem
ienvia_requisicao_driver(operacao,  index, buffer)
começa
        requisicao-operacao = operacao;
        requisicao-index = index;
        requisicao-id_share_bloco = __buffer;
        requisicao-pid = getpid();

        envia_mensagem_requisicao( requisicao-operacao, requisicao-index, requisicao-
id_share_bloco, requisicao-pid);
        retorna status;

final
```

Driver

```
        // Driver fica na ativa sempre, escutando
        escutar(sempre)
        começa
                escreva( Driver escutando);

                // REQUISICAO  da CID
                checar_nova_mensagem(cid)
                        if(novamensagem)
                                escreva(Chegou nova mensagem);

                confirmar_requisicao(cid)
                começa
                        caso 1:
                                /// Ler bloco bfd

                        caso 2:
                                // Ler outros blocos

                        caso 999: parar driver


                // REPOSTA PARA a CID
                envia_mensagem_resposta(status);
        final
```

FFOPEN

começa

verifica_caminho_arquivo(absoluto/relativo)

forma_caminho_absoluto(caminho_pcb+caminho_usuario)

começa_loop

envia_requisicao_driver(bfd)

envia _requisicao_bloco_driver(raiz)

verifica_token_caminho(bloco_raiz)

busca_proximo_bloco_no_bfd(index_raiz)

volta_loop

atualiza(listaaa)

atualiza(pcbdescritores)

retorna descritor

final