Relatório 5 – Laboratório de Sistemas Operacionais

Nome: Bruno Villas Boas da Costa,	**RA: 317527**
Henriete Hossri Iamarino,	**RA : 317489**
Mariana Landulpho Martini	**RA: 317179**

Introdução

Para facilitar a implementação, novas funções foram criadas, pois tiveram seu uso muito requisitado no decorrer do programa. Estas funções estão a seguir, juntamente com a explicação da sua utilidade:

- **fs_writeFat() :** Função utilizada para escrever toda a FAT no disco.
- **fs_writeDir():** Função utilizada para escrever todo o diretório no disco.
- **fs_ffree_cluster():** Função utilizada para encontrar o primeiro agrupamento livre da FAT.
- **fs_isFormated():** Função utilizada para verificar se o disco está formatado.
- **fs_write_sector(int cluster, char *buffer):** Função utilizada para escrever o parâmetro cluster nos setores do disco.
- **fs_read_sector(int cluster, char *buffer):** Função utilizada para ler o parâmetro cluster dos setores do disco.

Com o auxílio destas funções foi possível implementar todas as funcionalidades apresentadas nos dois grupos de funções.

Grupos de Funções

1. GERÊNCIA DE ARQUIVOS:

- **fs_init():** Esta função é chamada ao iniciar o sistema. Ela lê a FAT e o diretório do disco para a memória e verifica se o disco passado como parâmetro está formatado. Caso não esteja, formate.
- **fs_format():** Esta função primeiramente inicializa a FAT setando para '3' todos os agrupamentos que pertencem a FAT e depois inicializa o agrupamento do diretório setando '4' para ele. Logo após, inicializa todos os agrupamentos como agrupamentos

livres, para isso, como especificado no projeto, atribuímos '1' para representar isso. Da mesma maneira, no diretório é atribuído espaço livre para ele todo.

➔ **fs_free():** Essa função percorre toda a FAT à procura de espaço livre para armazenar o arquivo.

➔ **fs_list(char *buffer, int size):** Essa função varre todo o diretório, procurando por espaços que estejam sendo utilizados. Caso encontre, ele lista os arquivos seguidos de seu zx por dois 'tabs' no buffer.

➔ **fs_create(char* file_name):** Primeiramente a função verifica se o nome digitado não é superior a 25 caracteres que é o limite para o nome do arquivo, logo em seguida percorre o diretório procurando se o nome do arquivo a ser criado já existe. Caso as condições citadas tiverem sido satisfeitas, procura no diretório a primeira posição livre, caso ele não esteja cheio, ele seta no diretório que essa posição agora está ocupada, atribui o nome ao arquivo, procura o primeiro agrupamento livre da FAT, e escreve na FAT.

➔ **fs_remove(char *file_name):** Essa função inicialmente percorre o diretório procurando se o nome do arquivo passado como parâmetro existe. Se ele existir ele seta o valor de uso para livre, e vai atribuindo livre para toda continuação do arquivo (caso ele esteja ocupando mais do que um espaço na FAT), até encontrar o número '2' que nesse caso representa final do arquivo.

2. ESCRITA E LEITURA DE ARQUIVOS

➔ **fs_open (char *file_name, int mode):**

- Modo Leitura: A função procura no diretório se o nome passado como parâmetro existe ou não, e abre o arquivo no modo leitura.
- Modo Escrita: A função remove esse arquivo como o nome passado como parâmetro (através do fs_remove) e o cria novamente (através do fs_create). Logo em seguida ele procura no diretório se o arquivo existe e coloca no modo de escrita.

➔ **fs_close(int file):** Essa função verifica se o arquivo existe e se ele está aberto. Se ele e atende às duas verificações ele fecha o arquivo, caso contrário ele avisa o erro.

➔ **fs_write(char *buffer, int size, int file):** Nessa função, primeiramente verifica se o arquivo está aberto, e se está aberto para modo de escrita. Se essas condições forem satisfeitas, ele vai escrevendo os bytes pedidos até que o tamanho do cluster chegue ao seu limite, então ele procura outro cluster livre e continua escrevendo os bytes pedidos até terminarem, ou não houver mais espaço na FAT.

➔ **fs_read(char *buffer, int size, int file):** Essa função checa se o arquivo está aberto, e se está aberto para modo de leitura. Caso a checagem esteja correta, se o tamanho passado como parâmetro para ler for menor que o tamanho do arquivo que está

carregado, ele somente lê e passa o tamanho de bytes que leu. Senão ele vai buscando a continuação do arquivo até que o tamanho de bytes pedidos seja satisfeito ou o arquivo termine. Sempre retornando a quantidade de bytes lido.