Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Eléctrica

Microsistema de Archivos

Profesor: Gunnar Wolf

Semestre 2024-2

Alumnos:

- Barrios Michelle
- Zenón Plata Andrea Sofía

Fecha de entrega: 19/05/2024

En este documento se explicará el funcionamiento de cada una de las funciones y las bibliotecas que se utilizaron para la realización del proyecto.

En cuanto a las bibliotecas, se utilizan las siguientes:

- stdio.h: Es la biblioteca estándar en C que permite utilizar funciones de entrada y de salida.
- stdlib.h: Es otra de las bibliotecas estándar en C y permite utilizar funciones para manipular memoria, convertir tipos, control de procesos, etc.
- string.h: Permite utilizar funciones para manipular cadenas de caracteres.
- stdint.h: Proporciona definiciones de tipos de datos enteros de tamaño fijo, como uint32_t y uint8_t.
- time.h: Permite utilizar funciones para manipular el tiempo.
- unistd.h: Permite acceso a funciones del sistema operativo como lo son sleep y getop.
- sys/stat.h: Proporciona definiciones de estructuras y funciones para obtener información sobre archivos y directorios.

1. Comenzando con la función leer_suerbloque:

```
void leer_superbloque(const char *fiunamfs_img_path, Superbloque *sb)
```

```
{



    FILE *file = fopen(fiunamfs_img_path, "rb");

    if (!file)

    {

        perror("No se pudo abrir el archivo");

        exit(EXIT_FAILURE);

    }



    fseek(file, 0, SEEK_SET);

    uint8_t superbloque[SUPERBLOQUE_TAMANO];

    fread(superbloque, 1, SUPERBLOQUE_TAMANO, file);



    memcpy(sb->nombre_fs, superbloque, 8);
```

```
    sb->nombre_fs[8] = '\0';

    memcpy(sb->version, superbloque + 10, 5);

    sb->version[5] = '\0';

    printf("\n\033[1m----------Microsistema de
archivos----------\033[0m\n");



    printf("%-20s: %s\n", "Nombre FS", sb->nombre_fs);

    printf("%-20s: %s\n", "Version FS leida", sb->version);


printf("\033[1m-----------------------------------------\033[0m\n");



    if (strcmp(sb->nombre_fs, "FiUnamFS") != 0)

    {

        fclose(file);

        fprintf(stderr, "El sistema de archivos no es FiUnamFS\n");

        exit(EXIT_FAILURE);

    }



    if (strcmp(sb->version, "24-2") != 0)

    {

        fclose(file);

        fprintf(stderr, "Versión del sistema de archivos no
compatible\n");

        exit(EXIT_FAILURE);

    }
```

```
    sb->tamano_cluster = *(uint32_t *)(superbloque + 40);

    sb->numero_clusters_directorio = *(uint32_t *)(superbloque + 45);

    sb->numero_maximo_clusters = *(uint32_t *)(superbloque + 50);

    sb->directorio_inicio = sb->tamano_cluster;

    sb->directorio_tamano = sb->numero_clusters_directorio *
sb->tamano_cluster;

    sb->entrada_directorio_tamano = ENTRADA_DIRECTORIO_TAMANO;



    fclose(file);

}
```

Esta función lee el superbloque (como su nombre lo indica) del sistema de archivos FiUnamFS desde el archivo de imagen.

Un superbloque es una estructura de datos que tiene información importante sobre el sistema de archivos, como el nombre del sistema, la versión, el tamaño del cluster, entre otras. La función lee la información del archivo y la almacena en una estructura llamada “Superbloque”.

2. Función imprimir fecha

```
void imprimir_fecha(const char *fecha_raw, char *fecha_formateada) {

    struct tm tm_fecha;

    memset(&tm_fecha, 0, sizeof(struct tm));



    sscanf(fecha_raw, "%4d%2d%2d%2d%2d%2d",

           &tm_fecha.tm_year, &tm_fecha.tm_mon, &tm_fecha.tm_mday,

           &tm_fecha.tm_hour, &tm_fecha.tm_min, &tm_fecha.tm_sec);



    tm_fecha.tm_year -= 1900;
```

```
    tm_fecha.tm_mon -= 1;



    time_t tiempo = mktime(&tm_fecha);



            strftime(fecha_formateada,    20,    "%Y-%m-%d    %H:%M:%S",
localtime(&tiempo));

}
```

Esta función agarra la cadena de caracteres, que es una fecha, en un formato específico y la convierte en un formato más claro. Utiliza funciones de la biblioteca estándar para analizar la cadena de caracteres y poderla resetear según un formato determinado.

3. Funcion listar archivos

```
void    listar_archivos(const    char    *fiunamfs_img_path,    uint32_t
directorio_inicio,      uint32_t      directorio_tamano,      uint32_t
entrada_directorio_tamano) {

    FILE *file = fopen(fiunamfs_img_path, "rb");

    if (!file) {

        perror("No se pudo abrir el archivo");

        exit(EXIT_FAILURE);

    }



    fseek(file, directorio_inicio, SEEK_SET);

    uint8_t *directorio = malloc(directorio_tamano);

    fread(directorio, 1, directorio_tamano, file);



          for   (uint32_t   i   =   0;   i   <   directorio_tamano;   i   +=
entrada_directorio_tamano) {
```

```
        uint8_t *entrada = directorio + i;
        char nombre_archivo[15];
        memcpy(nombre_archivo, entrada + 1, 14);
        nombre_archivo[14] = '\0';
        for (int j = 0; j < 14; j++) {
            if (nombre_archivo[j] == '-') {
                nombre_archivo[j] = '\0';
                break;
            }
        }

        if (strlen(nombre_archivo) == 0 || nombre_archivo[0] == '.') {
            continue;
        }

        uint32_t tamano_archivo = *(uint32_t *)(entrada + 16);
        uint32_t cluster_inicial = *(uint32_t *)(entrada + 20);
        char fecha_creacion[15], fecha_modificacion[15];
        memcpy(fecha_creacion, entrada + 24, 14);
        fecha_creacion[14] = '\0';
        memcpy(fecha_modificacion, entrada + 38, 14);
        fecha_modificacion[14] = '\0';

                                    char    fecha_creacion_formateada[20],
fecha_modificacion_formateada[20];
```

```
        imprimir_fecha(fecha_creacion, fecha_creacion_formateada);

                                    imprimir_fecha(fecha_modificacion,
fecha_modificacion_formateada);



               if  (tamano_archivo  >  0  &&  cluster_inicial  >  0  &&
strlen(fecha_creacion) > 0 && strlen(fecha_modificacion) > 0) {

            printf("\033[1m---> Archivo:\033[0m %s\n", nombre_archivo);

                  printf("Tama%co                          %u\n",  164,
tamano_archivo);

            printf("Cluster Inicial         %u\n", cluster_inicial);

                    printf("Fecha  de  Creaci%cn           %s\n",  162,
fecha_creacion_formateada);

                      printf("Fecha  de  Modificaci%cn     %s\n",  162,
fecha_modificacion_formateada);



            printf("\n\n");

        }

    }



    free(directorio);

    fclose(file);

}
```

Esta función muestra todos los archivos que hay en el sistema de archivos FiUnamFS. Lee el directorio del sistema de archivos desde el archivo de imagen y recorre las entradas del directorio una por una. Imprime la siguiente información: el nombre del archivo, el tamaño, el cluster inicial y las fechas de creación y modificación respectivamente.

4. Función listar nombres archivos

```
void listar_nombres_archivos(const char *fiunamfs_img_path, uint32_t directorio_inicio, uint32_t directorio_tamano, uint32_t entrada_directorio_tamano)

{

    FILE *file = fopen(fiunamfs_img_path, "rb");

    if (!file)

    {

        perror("No se pudo abrir el archivo");

        exit(EXIT_FAILURE);

    }



    fseek(file, directorio_inicio, SEEK_SET);

    uint8_t *directorio = malloc(directorio_tamano);

    fread(directorio, 1, directorio_tamano, file);



          for (uint32_t i = 0; i < directorio_tamano; i += entrada_directorio_tamano)

    {

        uint8_t *entrada = directorio + i;

        char nombre_archivo[15];

        memcpy(nombre_archivo, entrada + 1, 14);

        nombre_archivo[14] = '\0';



            if (strlen(nombre_archivo) > 0 && strcmp(nombre_archivo, "--------------") != 0 && strcmp(nombre_archivo, "..............") != 0)

        {
```

```
            printf("%s\n", nombre_archivo);

        }

    }



    free(directorio);

    fclose(file);

}
```

Esta función se parece a la anterior, sin embargo, en esta se muestran los espacios vacíos con ############

5. Función copiar a sistema local

```
void copiar_a_sistema_local(const char *fiunamfs_img_path, uint32_t
directorio_inicio, uint32_t directorio_tamano, uint32_t
entrada_directorio_tamano, uint32_t tamano_cluster)

{

                printf("\n-----------------------------------Arhivos
existentes-----------------------------------\n\n");

         listar_nombres_archivos(fiunamfs_img_path, directorio_inicio,
directorio_tamano, entrada_directorio_tamano);



    char nombre_archivo_destino[256];

    char ruta_destino[256];

    printf("\nIngrese el nombre del archivo a copiar: ");

    scanf("%s", nombre_archivo_destino);

    printf("\nIngrese la ruta de destino del sistema *Incluir el nombre
del archivo*:  ");

    scanf("%s", ruta_destino);
```

```
    FILE *fiunamfs = fopen(fiunamfs_img_path, "rb");

    if (!fiunamfs)

    {

        perror("No se pudo abrir el archivo");

        exit(EXIT_FAILURE);

    }



    fseek(fiunamfs, directorio_inicio, SEEK_SET);

    uint8_t *directorio = malloc(directorio_tamano);

    fread(directorio, 1, directorio_tamano, fiunamfs);



    int archivo_encontrado = 0;



          for   (uint32_t   i   =   0;   i   <   directorio_tamano;   i   +=
entrada_directorio_tamano)

    {

        uint8_t *entrada = directorio + i;

        char nombre_archivo[15];

        memcpy(nombre_archivo, entrada + 1, 14);

        nombre_archivo[14] = '\0';



        uint32_t tamano_archivo = *(uint32_t *)(entrada + 16);

        uint32_t cluster_inicial = *(uint32_t *)(entrada + 20);



            if  (strcmp(nombre_archivo,  nombre_archivo_destino)  ==  0  &&
tamano_archivo > 0)
```

```
        {
            archivo_encontrado = 1;

                        uint32_t posicion_inicial = cluster_inicial *
tamano_cluster;
            fseek(fiunamfs, posicion_inicial, SEEK_SET);
            uint8_t *datos_archivo = malloc(tamano_archivo);
            fread(datos_archivo, 1, tamano_archivo, fiunamfs);

            if (ferror(fiunamfs))
            {
                perror("Error leyendo el archivo");
                free(datos_archivo);
                fclose(fiunamfs);
                return;
            }

            FILE *archivo_local = fopen(ruta_destino, "wb");
            if (!archivo_local)
            {
                perror("No se pudo crear el archivo local");
                free(datos_archivo);
                fclose(fiunamfs);
                return;
            }
```

```
            fwrite(datos_archivo, 1, tamano_archivo, archivo_local);
            fclose(archivo_local);
            free(datos_archivo);
                printf("\nARCHIVO '%s' COPIADO CON EXITO A '%s'.\n",
nombre_archivo_destino, ruta_destino);
            break;
        }
    }

    if (!archivo_encontrado)
    {
             printf("No se encontró el archivo '%s' en FiUnamFS.\n",
nombre_archivo_destino);
    }

    free(directorio);
    fclose(fiunamfs);
}
```

Esta función permite copiar un archivo del sistema de archivos FiUnamFS al sistema local. El usuario proporciona el nombre del archivo a copiar y la ruta de destino donde se copiará el archivo. La función busca el archivo en el directorio, lo lee del archivo de imagen, y luego lo escribe en el sistema de archivos local.

6. Función limpiar Pantalla E

```
void limpiarPantallaE()
{
```

```
    printf("\nPresione Enter para continuar...");

    while (getchar() != '\n')

        ;

    getchar();

    system("cls || clear");

}
```

Esta función espera a que el usuario ingrese un Enter para limpiar la pantalla de la consola. Es simplemente para pausar la salida y limpiar el área.

7. Función limpiar pantalla S

```
void limpiarPantallaS()

{

    printf("Redireccionando...\n");

    sleep(1);                // Espera 3 segundos

    system("cls || clear"); // Limpia la pantalla

}
```

Esta función simplemente muestra un mensaje al usuario por consola para indicar que se está redireccionando y posteriormente limpia la pantalla luego de que pasa un segundo.

8. Función main

```
int main()

{

    const char *fiunamfs_img_path = "fiunamfs.img";

    Superbloque sb;
```

```
    leer_superbloque(fiunamfs_img_path, &sb);


    while (1)
    {
        printf("\n\t=== Menu principal ===\n");
        printf("1. Mostrar archivos\n");
        printf("2. Copiar archivo de FiUnamFs al sistema\n");
        printf("0. Salir\n");


        char opcion[3];
        printf("\n\nSeleccione una opci%cn: ", 162);
        scanf("%s", opcion);
        limpiarPantallaS();


        if (strcmp(opcion, "1") == 0)
        {
                printf("\n-------------------------------------Listado de
archivos-------------------------------------\n\n");


                listar_archivos(fiunamfs_img_path, sb.directorio_inicio,
sb.directorio_tamano, sb.entrada_directorio_tamano);
            limpiarPantallaE();
        }
        else if (strcmp(opcion, "2") == 0)
        {
```

```
                            copiar_a_sistema_local(fiunamfs_img_path,
sb.directorio_inicio,                               sb.directorio_tamano,
sb.entrada_directorio_tamano, sb.tamano_cluster);

            limpiarPantallaE();

        }

        else if (strcmp(opcion, "0") == 0)

        {

            printf("Terminando programa...\n");

            break;

        }

        else

        {

                printf("\n\nOpci%cn no v%clida. Por favor, intente de
nuevo.\n", 162, 160);

        }

    }



    return 0;

}
```

Esta es la función principal del programa desde donde se mandan a llamar a las demás funciones. Primero lee el superbloque del sistema de archivos y luego muestra un menú por consola en el que da a elegir entre opciones para realizar.