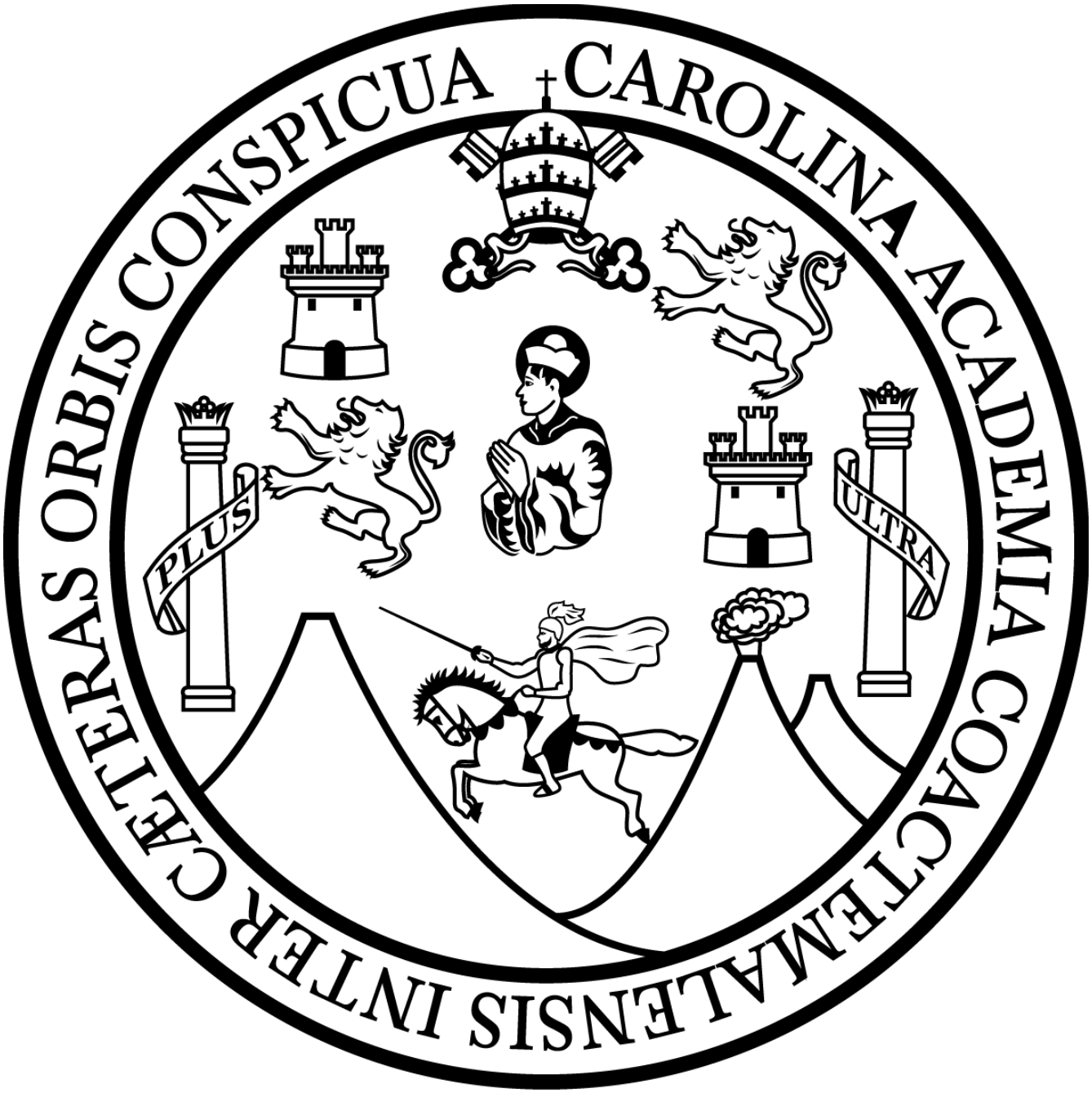
PROYECTO 1
MANEJO E IMPLEMENTACION DE ARCHIVOS
ESCUELA DE VACACIONES DICIEMBRE 2017
201403703

MANUAL TECNICO



Esta aplicación es un software libre el cual consiste en un sistema de archivos HADOOP, el cual se maneja como un sistema distribuido, Un sistema de archivos distribuido está diseñado para almacenar una gran cantidad de información para proveer accesos a muchos clientes distribuidos. Existe un largo número de sistemas distribuidos que solucionan este problema de diferentes formas.

HDFS (Hadoop Distributed File System), es un sistema de archivos distribuido diseñado para almacenar una gran cantidad de información (terabytes o petabytes) y provee un fácil acceso a la información. Los archivos son almacenados de manera redundante entre múltiples máquinas para asegurar la durabilidad y alta disponibilidad sobre aplicaciones en paralelo. El diseño de HDFS está basado en el diseño de GFS (Google File System), el sistema de archivos de Google.
HDFS es un sistema de archivos basado en bloques: los archivos individuales son divididos en bloques de un tamaño fijo. Estos bloques son almacenados alrededor de un cluster de una o varias máquinas con capacidad de almacenamiento. Las maquinas individuales en el cluster son llamadas DataNodes. Un archivo puede ser hecho de varios bloques, y ellos no necesariamente están almacenados en la misma maquina; las maquinas objetivo que almacenan cada bloque son escogidas al azar. El acceso a unos archivos puede requerir la cooperación de múltiples máquinas.

ESPECIFICACIONES TECNICAS:

El programa fue desarrollado en el lenguaje de C
Utilizando el IDE de Netbeans
Peso aproximado es de 332KB el cual lo hace un programa ligero
Es necesario al menos 50 GB de disco duro para poder manejar grandes cantidades de informacion si se desea.

HEADERS UTILIZADOS

manejoarchivosdirectorios_201403703.h:
Descripcion:
Se manipula directamente con todas las funcionalidades internas del sistema de archivos.

Metodos Utilizados:

```
void crearCarpeta(char path[250], char ruta[128]);
int revisarTablaNombres();
int buscarCarpeta();
int buscarEspacioVacio();
void copiarCadena();
int crearTableName();
int buscarPadre();
void buscarCoincidencias();
int  buscarCoincidenciasEnCarpetas();
int armarRuta();
int compararCadenaSubString();
void generarReporteArchivo();
void buscarCoincidenciasArchivo();
void generandoReporteArchivos();
int armarRutaArchivo();
void reporteNodos();
```

```
void reporteTablaNombres();
int revisarTablaNombresEliminar();
void eliminarDirectorio();
void copiarCarpeta();
```

manejodisco_201403703.h:
Descripcion:
En este header con su .c ayuda a la manipulacion del disco Maestro y a sus nodos.

Metodos Utilizados:

```
void crearDisco(char nombre_disco[100], char ruta[128], int cantidad_nodos, int tamanio_nodo);
void crearDirectorios(char path[128]);
void crear_mas_nodos(char path[128], char nombreMaestro[100]);
void eliminarDisco();
```

structs_201403703.h:
Descripcion:
Se declararon las estructuras que se usarian para tal sistema de archivos

Metodos Utilizados:

```
typedef struct MBR{
    int mbr_cantidad_nodos; //contiene la cantidad de DataNodes
    char mbr_numero_magico[10];
    int mbr_tabla_nodos; //contiene donde inicia la tabla de nodos
    int mbr_tabla_nombres; //contiene el byte donde inicia la tabla de nombres
} MBR;


typedef struct TABLA_NODOS{
    char nodo_ruta_nodo[128]; // contiene la ruta del archivo del Datanode
    int nodo_tamanio_nodo; // Contiene el tamanio actual del DataNode sin metadatos
    int nodo_tamanio_nodo_m; // Contiene el tamanio actual del DataNode con metadatos.
}TABLA_NODOS;

typedef struct TABLA_NOMBRES{
    char nombre_nombre[20]; //contiene el nombre del archivo o directorio
    int nombre_tipo; // 1 archivo  0 carpeta
    char nombre_fecha_creacion[250]; // fecha de la creacion
    int nombre_bloque_inicial; // numero de la estructura donde inicial el bloque de datos que contiene
la informacion del archivo o numero de la estructura de tabla de nombres que almacena la carpeta
    int nombre_datanode; //Numero de DataNode donde se almacena el bloque inicial, nulo si es una
carpeta
    int nombre_padre; //numero de la estructura de la tabla de nombres del directorio padre que contiene
esta carpeta o archivo
    int nombre_estado; //estado del registro (ocupado o libre)
}TABLA_NOMBRES;
```

```
typedef struct TABLA{
   char nombre_carpeta[20];
   int tabla_padre;
   TABLA_NOMBRES structura_tabla_nombre[9];
}TABLA;

typedef struct BLOQUE_DATO{
   char bd_data[51]; // contiene la informacion del archivo
   int bd_numero; // numero de bloques de datos
   int bd_siguiente; //numero de bloque siguiente que contiene la informacion
   int bd_datanode; //numero de DataNode donde se encuentra localizado el siguiente bloque
   int bd_estado; // estado del bloque (ocupado o libre)
}BLOQUE_DATO;

typedef struct colaPadres{
   char *padre;
   struct colaPadres *siguiente;
}colaPadres;

typedef struct colaDiscos{
   FILE * disco;
   struct colaDiscos * siguiente;
}colaDiscos;
```

Se utilizaron librerías como:
#include <stdio.h>
que significa "standard input-output header" (cabecera estándar E/S), es el archivo de cabecera que contiene las definiciones de las macros, las constantes, las declaraciones de funciones de la biblioteca estándar del lenguaje de programación C para hacer operaciones, estándar, de entrada y salida, así como la definición de tipos necesarias para dichas operaciones.

#include <stdlib.h>
(std-lib: standard library o biblioteca estándar). Es el archivo de cabecera de la biblioteca estándar de propósito general del lenguaje de programación C. Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.

#include <string.h>
es un archivo de la Biblioteca estándar del lenguaje de programación C que contiene la definición de macros, constantes, funciones y tipos y algunas operaciones de manipulación de memoria.

#include <time.h>
Esta librería se utilizo para darle el tiempo de creacion a cada nodo y tabla de nombre.