****

**Universidad Tecnológica Centroamericana**

**Laureate International Universities**

**Organización de Archivos**

Dr. Carlos Roberto Arias

***Avance de Proyecto***

**Sistema Gestor de Cobros**

*Presentado por:*

Guillermo Enrique Rodriguez Mazzoni

Jorge Alejandro Caballero Murillo

CTAs# 11041303 y 11111052

Lunes 18 de Febrero de 2012

Tegucigalpa, Honduras MDC

Índice

Introducción 3

Objetivos 3

Resultados 3

Tiempos de respuesta: (MacBook Pro i7 16GB) 3

Marco Teórico 5

Avail List 5

Archivos de Registros de Longitud Fija 5

Mantenimiento (Reportes Sin Índices) 5

Índices 5

Propiedades de un archivo de Índices 6

Reportes con Índices 6

Árbol B 6

Propiedades de un Árbol B 6

UML 7

Implementación 8

Manual de Usuario 8

Mantenimiento de los registros 8

Reportes 9

Llamadas por cliente por rango de fecha 10

Generar todas las facturas por rango de fecha 10

Rendimiento 11

Conclusiones 12

Dificultades 12

Recomendaciones 12

# Introducción

El siguiente proyecto se basa en la búsqueda de la optimización, del rendimiento de accesos al disco duro y otros medios de almacenamiento secundarios como también el rendimiento del uso de memoria primaria. Para medir los niveles de tiempo que le toma al programa trabajar a la par con el disco duro se utilizan tres estructuras de datos diferentes: Directamente en los archivos, haciendo uso de estructuras de índices, y la implementación de Arboles B.

# Objetivos

* Implementar los conocimientos adquiridos en el aula de Organización de Archivos, en un programa que evalué el desempeño del disco duro.
* Establecer correspondencia de datos entre las tres diferentes estructuras de datos.
* Hacer uso de las estructuras de índices para crear accesos directos al disco duro.
* Implementar Arboles B, haciendo más eficaz el tiempo de lectura é escritura en el disco duro.
* Generar Reportes haciendo uso de las tres estructuras de datos, y a su vez almacenarlas en un archivo .txt (Disco Duro).
* Mantener la integridad de los datos, mediante validaciones que se ejecutan en las entradas del programa.

# Resultados

El programa demuestra mejor respuesta en cuanto a tiempo cuando se utiliza la implementación de Arboles B a la hora de generar reportes. Ya que los tiempos de búsqueda son más eficiente, le toma muchísimo más tiempo al programa generar reportes haciendo uso de índices, y una eternidad cuando no se utiliza ni índices.

### Tiempos de respuesta: (MacBook Pro i7 16GB)

Arboles B:

Índices: 20 segundos por Cliente

Sin Índices: 28 segundos por Cliente

**En este proyecto se toma en cuenta las siguientes operaciones:**

* Mantenimiento
  + Clientes
    - Agregar
    - Buscar
    - Modificar
    - Borrar
    - Listar
  + Teléfonos
    - Agregar
    - Buscar
    - Modificar
    - Borrar
    - Listar
  + Tipos de Teléfonos
    - Agregar
    - Buscar
    - Modificar
    - Borrar
    - Listar
  + Tarifas
    - Agregar
    - Buscar
    - Modificar
    - Borrar
    - Listar

Las operaciones de mantenimiento se incorporaron a las tres estructuras de datos ya antes mencionadas, en las cuales mantenimiento pasaba a ser una parte fundamental en cada una de las diferentes estructuras, ya se debía mantener la correspondencia de datos.

# Marco Teórico

El proyecto se presenta en C++, compilado con g++. Se utilizaron los siguientes sistemas operativos:

* Ubuntu 12.04 LTS en una máquina virtual de 64-bit en un host MacBook Pro con el agente virtualizador Parallels Desktop.
* Ubuntu 12.10 LTS instalado como host de 32-bit.

En la primera presentación el programa se alimenta de archivos de texto con registros de tamaño fijo y campos de tamaño fijo. A medida que el proyecto y la clase avancen este método de alimentación ira siendo más eficiente en cuanto a acceso a memoria primaria tanto como secundaria.

En la segunda entrega del avance del proyecto, el objetivo era haber implementado las estructuras de índices vistas en el aula. Logrando a su vez accesos directos al disco duro, mayor tiempo de respuesta

### Avail List

Es un archivo que maneja el siguiente RRN disponible para inserción (En caso de ARLF) o el offset y caracteres a partir del offset disponibles para la escritura (en caso de ARLV). Consiste en “marcar” el registro en cuestión como “inactivo” con un “\*” o un carácter similar. El proceso varia para ARLV & ARLF. El Avail List se puede manejar en memoria por medio de cursores, pilas, vectores, ect.

### Archivos de Registros de Longitud Fija

Si el availist está vacío, se agrega el número de caracteres (ej. 123) al final del archivo que estamos manejando. Si el Avail List tiene algo en ella (ej 5 o 1239), se inserta nuestro número de caracteres en el espacio disponible delineado en nuestro Avail List , sobre escribiendo lo que este en ese intervalo especificado y actualizamos el Avail List después de la sobre escritura.

### Mantenimiento (Reportes Sin Índices)

En las operaciones de mantenimiento se utilizó la librería de fstream para poder leer, editar, escribir en los archivos maestros que contienen los registros de Client, Phone, Calls. (Todo se manejó directo en el disco duro)

### Índices

Son estructuras “sencillas” que permiten que encontremos campos o registros de una manera más fácil y eficiente. El índice puede definirse como una entidad que opera con un procedimiento que acepta información acerca de cierto valor de atributo, como entrada, y provee como salida una información que permite la rápida localización del registro.

Para que la búsqueda de un elemento en un archivo de índices sea más eficiente y reducir el tamaño del archivo de índice, los campos referentes a la información asociada se almacena separadamente, con los cuales se tendrán que considerar una colección ante la llave y datos asociados, esta es a través de un apuntador.

### Propiedades de un archivo de Índices

* Su peso en el disco es menor al del archivo original de datos.
* No hay necesidad de reorganizar los datos originales ya que es más eficiente ordenar un archivo de índices.
* Su recorrido es de menos impacto para la memoria primaria, y podemos suponer que es posible montar el archivo de índices completo a la memoria.
* Permite efectuar más fácilmente las transferencias y eliminaciones de datos.

### Reportes con Índices

En la implementación para generar reportes utilizando índices se utilizaron las librerías <maps>, las cuales se personalizaron para hacer uso de mantenimiento. Se decidió utilizar mapas ya que los mapas ofrecen la opción de mantener un listado ordenando mediante una llave primaria y a medida que va creciendo o desintegrándose siempre lo mantendrá ordenado de menor a mayor. En el segundo parámetro del map “second” se ingresa el RRN de dicha llave primaria por ejemplo:

Client\_Map.insert(pair<string,string >(1080119930001, 202) );

También se utilizaron multimapas ya que ofrecen la ventaja de mapas, a esto agregado que la llave primaria se puede repetir y de igual manera siempre lo mantiene ordenado.

### Árbol B

En las [ciencias de la computación](http://es.wikipedia.org/wiki/Ciencias_de_la_computaci%C3%B3n), los árboles-B o B-árboles son [estructuras de datos de árbol](http://es.wikipedia.org/wiki/%C3%81rbol_(programaci%C3%B3n)) que se encuentran comúnmente en las implementaciones de [bases de datos](http://es.wikipedia.org/wiki/Bases_de_datos) y [sistemas de archivos](http://es.wikipedia.org/wiki/Sistemas_de_archivos). Son [árboles balanceados de búsqueda](http://es.wikipedia.org/wiki/%C3%81rbol_binario_de_b%C3%BAsqueda) en los cuales cada nodo puede poseer más de dos hijos.  Los árboles B mantienen los datos ordenados y las inserciones y eliminaciones se realizan en tiempo logarítmico amortizado.

### Propiedades de un Árbol B

* Orden m
* Cada página tiene un máximo de m hijos.
* Cada página interna tiene al menos [m/2] hijos.
* La raíz tiene al menos 2 hijos (excepto hojas).
* Todos los hijos están al mismo nivel.
* Una página no-hoja con r-descendientes tiene r -1 llaves.
* Una página hoja tiene al menos [m/2] -1 llaves y no más de m -1 llaves.

# UML

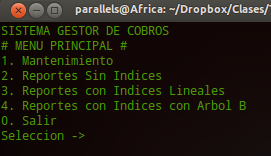
# Implementación

El código se presenta documentado para referencia de implementación.

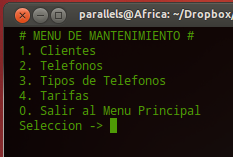
# Manual de Usuario

## Mantenimiento de los registros

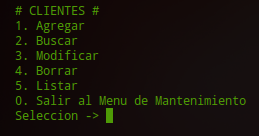
Al iniciar el programa se presentara un menú principal que consta con las siguientes opciones:



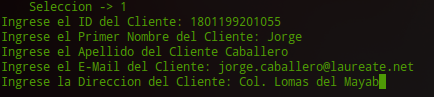
Al entrar al menú de mantenimiento vemos desplegadas las siguientes opciones con su desglose:



Al entrar al menú de Clientes y Teléfonos nos da las siguientes opciones:



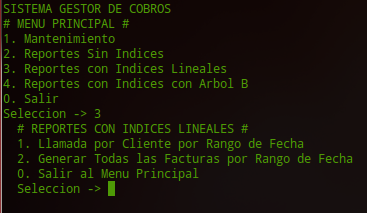
Para agregar un cliente se llena la siguiente información:



Al darle “Enter” se salvara al archivo siempre con el orden deseado y usando cualquier espacio que este disponible marcado por el AvailList.

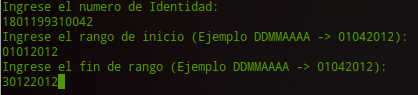
## Reportes

Para todos los 3 casos de los resportes el programa pregunta los mismos campos, para este ejemplo usaremos Reportes con Indices:



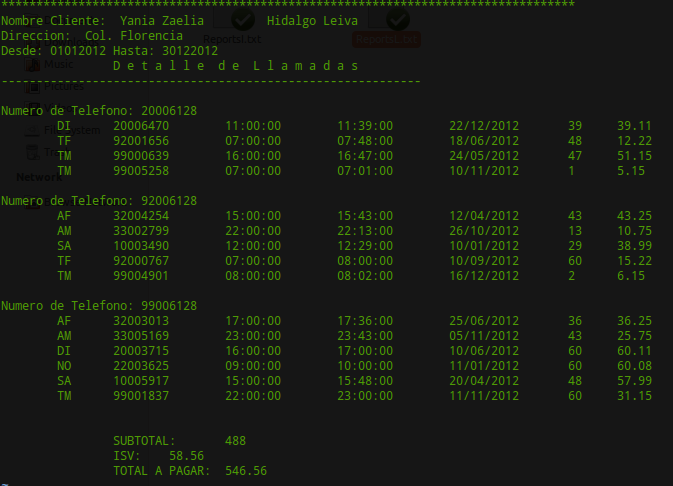
### Llamadas por cliente por rango de fecha

Al seleccionar la opcion 1 nos pregunta los siguientes campos:



Al presionar “Enter” generara el reporte de la persona con la ID 1801199310042 desde el 1/1/2012 hasta el 30/12/2012 y lo guardara en el folder /reports/ReportsI.txt

Al ver este documento de texto nos encontraremos con lo siguiente:



### Generar todas las facturas por rango de fecha

Al entrar a la opcion de generar las facrueas por rango de fecha nos encontraremos con las siguientes entradas:



# Rendimiento

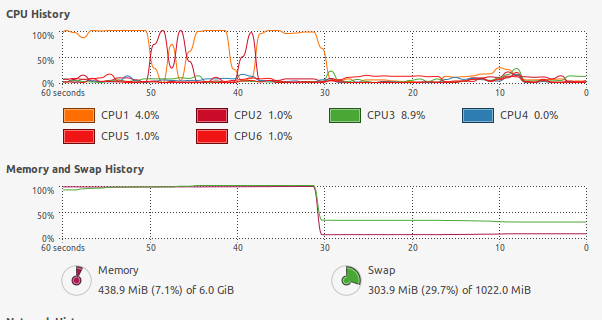
Luego de varias ejecuciones hemos conluido que el programa a pesar de su implementacion de indices y arboles, necesita demasiado recursos.

Este es un ejemplo de nuestra ejecucuion en una maquina virtual con la siguiente manera:

* 6GB de RAM
* 6 CPU de 2.8Ghz



Y esto es el Resource Manager cuando el programa fue “Killed” por la maquina por su uso excesivo de recursos:



# Conclusiones

Se comprendieron todos los conceptos adquiridos en el aula de Organización de Archivos, poniéndolos en práctica en el desarrollo del proyecto.

Se logró concientizar que el tiempo en acceder al disco duro es vital para el funcionamiento de un programa. Esto se ve influido dependiendo que estructura de datos estemos utilizando.

### Dificultades

* Una de las mayores dificultades del proyecto, fue a la hora de generar los archivos maestros los cuales almacenaban los registros. La correspondencia de datos debía ser igual ya que a la hora de referenciar datos o de utilizar las estructuras de índices o Arboles B había conflicto.

**Solución:** Generar los archivos maestros por partes e irlos unificando.

* Encontramos un grado de dificultad a la hora de implementar el método read ya que no nos devolvía lo que esperábamos (Objeto Client, Phone).

**Solución:** Se debía agregar un char más, para marcar el fin del string. Por otro lado los cálculos debían ser exactos de los RRN ya que eran de tamaño fijo.

### Recomendaciones

* Siempre tratar de afinar un concepto o herramienta aprendida en el pasado.
* Maps y Mulitmaps fueron de gran ayuda en este proyecto, ya que podíamos aplicar estructuras como Lista de Doblemente Enlazadas o Lista de Adyacencia.
* Siempre llevar las ideas a un bosquejo, antes de ponerlas en práctica.

*No creímos en un principio lograr llevar este proyecto a su finalidad en C++, Pero nos enorgullece decir que lo logramos al final del día.*