UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

LABORATORIO II

Paradigma Lógico Aplicado En La Elaboración De Un Sistema Operativo de Archivos

Paradigmas de Programación - 13310
Profesor Roberto González Ibáñez, PhD.

Byron Caices Lima

12 de Junio, 2023

Tabla de Contenidos

1. INTRODUCCIÓN

En el presente informe se abordará el problema propuesto en el enunciado de laboratorio 2 acerca del paradigma lógico aplicado en el lenguaje de programación SWI-Prolog. Se revisará una descripción del problema y del paradigma empleado además del diseño de la solución y aspectos de la implementación.

2. DESCRIPCIÓN DEL PROBLEMA

Se presenta como desafío la creación de un sistema operativo de archivos (con orientación al usuario) en donde se tenga un sistema al cual se le pueden añadir unidades o drives y a estas unidades pueden añadirse carpetas y archivos implementando operaciones tales como añadir/formatear unidad, añadir/borrar/renombrar/encriptar carpeta u archivo. Para esto se debe tener en consideración la existencia de una papelera para permitir operaciones como la de restaurar archivo o carpeta. Finalmente se apunta a simular la consola del sistema introduciendo las operaciones típicas como cd, md, dir, etc.

3. DESCRIPCIÓN DEL PARADIGMA Y CONCEPTOS APLICADOS

En el paradigma de programación lógica, se compone de hechos y reglas, en lugar de secuencias de instrucciones. Los predicados son declaraciones que pueden ser verdaderas o falsas y a menudo tienen variables. Estas variables pueden tomar valores de dominios específicos, que son los conjuntos de todos los posibles valores que las variables pueden tener. Las metas representan las consultas o cuestiones que se plantean al programa. Estas metas se resuelven utilizando los hechos y las reglas definidos en el programa, a través de un proceso llamado unificación. La unificación es el proceso de encontrar una asignación de valores a las variables de tal manera que dos términos lógicos se vuelvan idénticos. El backtracking es una estrategia de búsqueda que se utiliza para encontrar todas las soluciones posibles a una meta. Cuando el programa encuentra una solución, retrocede para buscar más soluciones. Si se encuentra en un punto muerto, retrocede aún más.

Además, el paradigma lógico a menudo utiliza la recursividad, una técnica en la que una regla puede hacer referencia a sí misma en su propia definición. Esto es especialmente útil para resolver problemas que tienen una estructura recursiva natural.

4. ANÁLISIS DEL PROBLEMA

Dados los requisitos funcionales específicos que se deben cubrir lo primero que se debe realizar para abordar este problema es la identificación de los TDA necesarios. Notamos que se tienen 5 principales elementos para interactuar en un sistema de archivos: Sistema, Unidades, Carpetas, Archivos y Usuarios. (Especificación de TDAs en Anexo 1)

A su vez, este sistema está compuesto por atributos como un nombre, usuarios registrados, rutas accesibles (paths), unidades, contenido de las unidades y ese contenido posee carpetas y archivos, y las carpetas pueden poseer más carpetas y archivos dentro. Al final, se asemeja a la estructura de árbol del Anexo 2. Sin embargo, implementar una estructura arbórea para la construcción del TDA puede ser un poco engorroso por lo que nos queda pensar en otra alternativa usando la estructura de datos más básica que nos ofrece Prolog: listas. Con esto se puede determinar que para manejar el sistema de archivos basta con trabajar con rutas; añadir una carpeta “Carpeta1” a una unidad “C:” no es más que agregar un path nuevo al sistema “C:/Carpeta1” y además agregar el TDA Carpeta al contenido del sistema para almacenar de alguna formas los TDAs que pueden ser trabajados en este. Sin embargo, no bastaría solo con agregar la ruta ya que nos faltaría saber, por ejemplo, la metadata de la carpeta o de un archivo como el usuario creador, fecha de creación, fecha de modificación y atributos de seguridad. También quedaría definir qué es borrar un archivo en mi sistema, y se llegó a la conclusión de que borrar una carpeta o archivo no es más que eliminar la ruta (location) de una carpeta o archivo del path del sistema para que así ese ítem eliminado se vuelva inaccesible e inmutable pero que de todas maneras siga estando almacenado en la unidad en que se encontraba antes de ser borrado, tiene lógica ya que un sistema por sí solo no podría tener la capacidad de almacenar archivos sino que estos son almacenados en una unidad. Luego, considerando que el paradigma lógico unifica variables (no las modifica) tendremos Sistemas los cuales no pueden ser modificados directamente si no que se tendrá que reconstruir o unificar el sistema completo si es que queremos realizar una modificación. Dados estos fundamentos del análisis del problema queda pasar al diseño de la solución.

5. Diseño De La Solución

Como se mencionó en el punto anterior el TDA Sistema se compone de atributos tales como la hora actual, usuarios registrados, unidades, papelera y paths. La forma en que se decidió representar el sistema corresponde a una lista con listas, exceptuando aquellos valores que por su naturaleza no es necesario que sean una sublista del sistema como, por ejemplo, el nombre de este, la fecha y la ruta actual del sistema, los que para el caso de mi implementación serán un string. Luego, ya las listas que componen al sistema principalmente serán la sublista de paths que contiene todas las rutas accesibles y mutables del sistema (una lista de strings), la sublista que contiene la papelera que contendrá aquellas rutas (strings) que se vuelven inaccesibles ya que su contenido fue eliminado y finalmente la sublista de unidades o drives tendrán cada una otra sublista del contenido del drive en donde se guardarán carpetas y archivos (a los que llamaremos items). Cada TDA Carpeta y Archivo tendrá el atributo location el cual se referirá a la ubicación del item…

6. Aspectos De La Implementación

7. Instrucciones De Uso

8. Resultados Y Autoevaluación

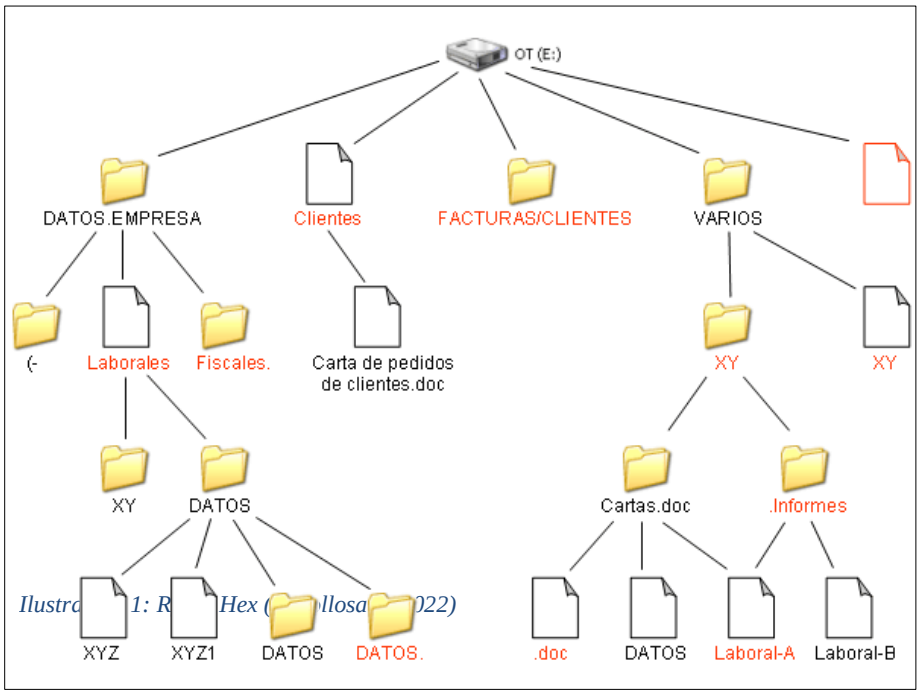
9. Conclusión

10. Referencias

11. Anexos

Anexo 1: Definición de los TDA´s

Anexo 2: Estructura de árboles en un sistema de archivos



Ilustra 1: R Hex (llosa 022)

Anexo 3: Ejemplos sobre las instrucciones de uso

Anexo 4:

Resumen resultados obtenidos (Listado con todos los requerimientos del proyecto)