

Paradigmas de la Programación

Informe Laboratorio nro. 2

Paradigma Lógico

*Profesor: Gonzalo Martínez.*

*Alumna: Karina Bustamante H.*

4 de junio de 2023

Tabla de contenido

[Introducción 2](#_Toc134312314)

[Descripción del problema 2](#_Toc134312315)

[Descripción del paradigma 3](#_Toc134312316)

[Análisis del problema 3](#_Toc134312317)

[Diseño de la solución 6](#_Toc134312318)

[Consideraciones de implementación 6](#_Toc134312319)

[Instrucciones de uso 7](#_Toc134312320)

[Resultados y evaluación 8](#_Toc134312321)

[Conclusiones 9](#_Toc134312322)

[Referencias 10](#_Toc134312323)

# Introducción

Para el desarrollo del laboratorio Nº2, de la asignatura de Paradigmas de la Programación, se nos solicita realizar una implementación bajo el paradigma lógico, el cual se basa en la lógica matemática y en la resolución de problemas mediante la deducción lógica. Los algoritmos a implementar deberán resolver el problema que corresponde al siguiente enunciado:

***“Crear un simulador de sistema operativo centrado específicamente en un sistema de archivos simplificado y los comandos que*** ***permiten operar sobre éste.”***

Para el desarrollo de esta implementación usaremos el lenguaje de programación, Prolog, con su editor virtual, (https://swish.swi-prolog.org/) y a través de la creación de una base de conocimientos que crearemos con predicados, reglas y clausulas, se buscará dar solución a los requerimientos funcionales solicitados, de manera tal, que la solución propuesta permita al usuario aplicar sobre este sistema de archivos ficticio, acciones como crear, buscar, listar, etc.

# 

# Descripción del paradigma

El paradigma lógico corresponde al paradigma de programación declarativo, puede usarse como la especificación de un problema en lugar de establecer los pasos necesarios para llegar a una solución. Se basa en la lógica booleana y no existen funciones ni retornos normales. En este paradigma, se describe el problema en términos de relaciones lógicas entre objetos y se utiliza la inferencia lógica para resolver el problema. Se declara una base de conocimiento que es un conjunto de hechos y reglas que describen las relaciones entre los objetos en un dominio de problema. El programador describe el objetivo o la pregunta que se quiere responder en términos de estas relaciones lógicas. Luego, el sistema de programación utiliza la lógica para deducir la respuesta al objetivo a partir de los hechos y reglas definidos en la base de conocimiento. Los hechos, son un tipo de cláusula que describe una relación entre uno o más término, estos hechos se asumen siempre verdaderos.

Los mecanismos básicos de este paradigma son:

* ***Unificación***: se refiere a la técnica de encontrar una asignación de valores a las variables en una expresión lógica que hace que la expresión sea verdadera. Si se encuentra una asignación de valores que haga que las expresiones sean iguales, la consulta se considera resuelta y se devuelve el resultado
* ***Backtraking automático***: Sucede cuando no se logra la unificación, el proceso da un paso atrás para probar otros caminos para lograr la unificación, si se encuentra una asignación de valores que haga que las expresiones sean iguales, la consulta se considera resuelta y se devuelve el resultado, True.
* ***Inferencia***: corresponde a la aplicación de reglas lógicas para derivar nuevas proposiciones a partir de proposiciones previas.

Si comparamos los paradigmas que ya conocemos, Funcional v/s Lógico, tenemos que el paradigma funcional se basa en la evaluación de expresiones matemáticas y la teoría de funciones puras y se debe lograr la inmutabilidad evitando así los cambios de estados, estos últimos, solo se consiguen generando nuevas estructuras de datos en lugar de modificar las existentes. Por otro lado el paradigma de programación lógico que se basa en la lógica matemática y la teoría de conjuntos, el cual se compone de un conjunto de hechos y reglas lógicas los que se usan para responder consultas mediante la inferencia y lo que se conoce como proceso de unificación. A través de ello, se buscan todas las respuestas posibles mediante la lógica de inferencia utilizando una programación declarativa.

De lo anterior podemos decir que, la programación lógica se basa en la lógica y la inferencia, mientras que la programación funcional se basa en la evaluación de expresiones matemáticas y la composición de funciones.

# Descripción del problema

El objetivo de este proyecto es desarrollar una simulación de un sistema de archivos utilizando el lenguaje de programación Prolog. Un sistema de archivos es una estructura jerárquica utilizada para organizar y almacenar información en un dispositivo de almacenamiento, como un disco duro. Para ello se debe desarrollar un conjunto de requerimientos funcionales basados en el paradigma lógico que permitan llevar a término esta simulación, donde un usuario pueda realizar acciones tales como crear y agregar unidades, usuarios, así como también crear, modificar y eliminar archivos o carpetas, según permisos asignados, generando rutas de acceso a los archivos y carpetas, cambiarse de ruta, crear usuarios y drivers, cambiarse de unidad, iniciar y cerrar sesión.

# Análisis del problema

El problema principal se centra en el diseño y la implementación de las reglas y hechos necesarios para simular un sistema de archivos eficiente y funcional en Prolog. Se deben tener en cuenta las siguientes consideraciones clave durante el análisis del problema:

***Jerarquía de directorios:*** El sistema de archivos debe permitir la creación y gestión de una estructura jerárquica de directorios. Cada directorio puede contener archivos y subdirectorios, y el sistema debe permitir la navegación fluida a través de esta estructura jerárquica.

***Creación y eliminación de archivos y directorios***: El sistema debe proporcionar la capacidad de crear y eliminar tanto archivos como directorios. Al crear un archivo, se debe especificar su nombre y ubicación en el sistema de archivos. Al eliminar un archivo o directorio, se deben eliminar todos sus contenidos y actualizar la estructura del sistema de archivos en consecuencia.

***Gestión de permisos y protección***: El sistema de archivos debe implementar un sistema de gestión de permisos que controle el acceso a los archivos y directorios. Se deben definir niveles de acceso, como lectura y escritura, y asegurarse de que se respeten los permisos definidos.

Al abordar estos aspectos clave, nuestra simulación de un sistema de archivos en Prolog proporcionará una herramienta funcional y útil para gestionar la organización y manipulación de archivos y directorios. En la sección siguiente, se detallará el diseño y la implementación del sistema de archivos en Prolog, cubriendo los predicados y reglas necesarios para lograr estos objetivos.

# Diseño de la solución

# El sistema de archivos para este laboratorio se diseñará e implementará utilizando una combinación de hechos y reglas en Prolog. A continuación, se detallan los principales elementos del diseño y la implementación:

* *Estructura de datos*: Los directorios se representarán como hechos que contienen un nombre y una lista de archivos y subdirectorios.

Los archivos se representarán como hechos que contienen un nombre y otros atributos relevantes como, extensión, usuario y fecha de creación.

Se utilizarán reglas para establecer relaciones entre los directorios y los archivos, permitiendo la navegación y la búsqueda.

* *Creación y eliminación*: Para crear un archivo o directorio, se utilizarán reglas que agreguen un nuevo hecho a la base de conocimientos, especificando el nombre y la ubicación en el sistema de archivos. Para eliminar un archivo o directorio, se utilizarán reglas que eliminen el hecho correspondiente y actualicen la estructura del sistema de archivos en consecuencia.
* Gestión de permisos y protección: Se utilizarán reglas que verifiquen los permisos de acceso antes de realizar operaciones en archivos o directorios. Se asignarán niveles de acceso mediante hechos que asocien los archivos y directorios con los usuarios y sus respectivos permisos.
* Operaciones adicionales: Para copiar un archivo o directorio, se utilizarán reglas que creen una nueva instancia con el mismo contenido en una ubicación diferente. Para cambiar el nombre de un archivo o directorio, se utilizarán reglas que actualicen el hecho correspondiente con el nuevo nombre. Este diseño y la implementación proporcionarán un sistema de archivos básico y funcional en Prolog. A medida que se avance en el proyecto, se podrán agregar características adicionales y refinar la implementación según sea necesario.

# Consideraciones de implementación

# La implementación de este laboratorio fue realizada a través del entorno virtual de prolog <https://swish.swi-prolog.org/> por ser desarrollado en un equipo computacional con sistema operativo macOS Monterrey. Por lo anterior se recomienda la utilización de este entorno virtual para realizar las pruebas de funcionamiento del código entregado.

# Instrucciones de uso

Para la ejecución de los algoritmos propuestos, debe abrir el archivo lab2\_13452929\_Bustamante.pl, seleccionar todo el contenido, copiar y pegar en el editor virtual de prolog, <https://swish.swi-prolog.org/>

Para hacer las pruebas de funcionamiento debe ingresar en el apartado para las consultas del intérprete de prolog, lo que se encuentra en el contenido del archivo, pruebas\_13452929\_Bustamante.pl

Para crear unidades de disco se debe considerar escribir la letra del disco junto con “:”, para una mejor visualización.

# Resultados y evaluación

A continuación se muestra la autoevaluación de los requerimientos funcionales requeridos por el laboratorio, y el grado de logro alcanzado para cada uno de ellos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RF Nº | Requerimiento funcional | Grado de alcance | Tipo o cantidad de pruebas | Éxitos | Fracasos | Razones de fallo |
| 1 | TDAs | 1 |  |  |  |  |
| 2 | system - constructor | 1 | 2 | 2 | 0 |  |
| 3 | system - addDrive | 1 | 2, uni. creadas | 2 | 0 |  |
| 4 | system - register | 1 | 3, si no existe usuario, se crea | 3 | 0 |  |
| 5 | system - login | 1 | 2 | 1 | 1 | Usuario no existe en lista de usuarios |
| 6 | system - logout | 1 | 2 | 2 |  |  |
| 7 | system - switch-drive | 1 | 2 | 1 | 1 | Unidad no existe en lista de unidades |
| 8 | system - mkdir | 0,75 | 4 | 1 | 3 | No asignaba bien los predicados. |
| 9 | system- cd |  |  |  |  |  |
| 10 | system - add-file | 0,05 | 3 | 1 | 2 | No logra mostrar agregar el archivo a pesar que en la ejecución paso a paso se ve que los agrega a la lista. |
| 11 | system - del |  |  |  |  |  |
| 12 | system - copy |  |  |  |  |  |
| 13 | system - move |  |  |  |  |  |
| 14 | system - ren |  |  |  |  |  |

**Tabla 1.-** Muestra los requerimientos funcionales y el nivel de logro.

# 

# Conclusiones

En este proyecto de laboratorio 2, de la asignatura de paradigmas de la programación, se ha logrado desarrollar con éxito parte de la simulación de un sistema de archivos utilizando el lenguaje de programación Prolog. A través del diseño y la implementación de reglas y hechos, se ha creado una herramienta para gestionar la organización, manipulación de archivos, unidades de disco y directorios.

Durante el análisis del problema, se identificaron los aspectos clave necesarios para simular un sistema de archivos completo. Se consideraron la jerarquía de directorios, la creación y eliminación de archivos y directorios, la gestión de permisos y protección, así como operaciones adicionales como la copia de archivos.

El diseño y la implementación se basaron en una estructura de datos utilizando hechos y reglas en Prolog. Se utilizaron hechos para representar directorios y archivos, y se establecieron relaciones mediante reglas para permitir la navegación. No se logró implementar reglas que cubrieran la totalidad de los requerimientos funcionales solicitados y pensados en el análisis de la solución correspondientes a eliminar archivos y directorios, ni tampoco para gestionar los permisos de acceso.

# 

# Referencias

* Swi-prolog, Reference manual. https://www.swi-prolog.org/pldoc/doc\_for?object=manual
* Campus Virtual, (2023). Paradigmas de Programación. https://uvirtual.usach.cl/moodle/