# Informe de Laboratorio II Paradigma Lógico: Prolog

Oscar Henríquez González

Junio de 2023

**Paradigmas de programación  
Profesor:** Gonzalo Martínez

## Tabla de contenido:

[Informe de Laboratorio II Paradigma Lógico Proglog 1](#_Toc133185792)

[Tabla de contenido: 2](#_Toc133185793)

[Introducción 4](#_Toc133185794)

[Descripción breve del problema 4](#_Toc133185795)

[Análisis del problema 5](#_Toc133185796)

[Diseño de la solución 5](#_Toc133185797)

[Aspectos de implementación 6](#_Toc133185798)

[Instrucciones de uso 6](#_Toc133185799)

[Resultados y autoevaluación: 7](#_Toc133185800)

[Conclusiones del trabajo 8](#_Toc133185801)

[Referencias 8](#_Toc133185805)

## Introducción

El siguiente laboratorio nos propone como desafío la implementación de un sistema de archivos, este tipo de software es fundamental en un sistema operativo para gestionar el almacenamiento de datos, se enfoca en la organización superficial de archivos y carpetas.

El sistema debe permitir múltiples usuarios, quienes pueden acceder a diferentes unidades y almacenar archivos de cualquier tipo y tamaño, considerando gran parte de las funciones de un sistema de archivos real como listar archivos, recorrer directorios, **e**tc.

## Descripción breve del problema

## El problema establecido será resuelto en el lenguaje de programación Prolog, dentro del paradigma lógico. En el paradigma lógico, se consideran las cláusulas y la unificación como los elementos más relevantes. Cabe mencionar que Prolog fue creado con el propósito de brindar un tratamiento algorítmico al lenguaje natural, y según Clocksin y Mellish, "Prolog es un lenguaje de programación que se utiliza para resolver problemas que involucran objetos y sus relaciones".

## Algunos conceptos clave del lenguaje Prolog incluyen:

## Backtracking: es un mecanismo que permite recordar las instancias de la ejecución donde un objetivo tiene diversas soluciones. Esto permite retroceder y continuar la ejecución utilizando otra solución.

## Unificación: es un procedimiento que combina dos predicados, permitiendo que los términos se emparejen.

## Hechos: son cláusulas que declaran relaciones entre objetos, estableciendo que ciertos elementos son verdaderos.

## Reglas: son otro tipo de cláusulas que contienen un cuerpo donde se llaman predicados. Si una regla no tiene cuerpo, se considera una cláusula tipo hecho.

## Además de los conceptos mencionados anteriormente, es importante destacar la relevancia de los tipos de datos abstractos (TDA), al igual que en el paradigma funcional. En el contexto de Prolog, los TDA se especifican de manera abstracta, teniendo propiedades lógicas y funcionales.

## Análisis del problema

## La construcción de un simulador de administrador de archivos en Prolog desde el punto de vista del paradigma lógico presenta varios aspectos interesantes. En primer lugar, el paradigma lógico se basa en la lógica de predicados, lo que permite modelar de manera natural las relaciones y reglas que gobiernan el sistema de archivos. Esto facilita la representación de la estructura jerárquica de directorios y archivos, así como las operaciones y consultas asociadas.

## En segundo lugar, el paradigma lógico se centra en la inferencia lógica y el razonamiento automatizado. Esto significa que, en lugar de especificar explícitamente los pasos para realizar una operación, como en otros paradigmas, se definen reglas lógicas y se realizan consultas para obtener respuestas. En el simulador de administrador de archivos, esto permite que el sistema realice deducciones y derive el resultado de una operación en función de las reglas y hechos definidos.

## Por último, el paradigma lógico fomenta la programación declarativa, donde se describe qué se quiere lograr en lugar de como lograrlo. Esto permite abstraerse de los detalles de implementación y centrarse en la lógica subyacente del sistema de archivos. Además, la capacidad de hacer consultas permite explorar y obtener información de manera interactiva, lo que facilita la interacción con el simulador y la realización de pruebas y análisis.

## En resumen, el paradigma lógico ofrece una base sólida para construir un simulador de administrador de archivos en Prolog, aprovechando la representación de relaciones y reglas lógicas, el razonamiento automatizado y la programación declarativa. Estas características permiten modelar el sistema de archivos de manera natural, realizar inferencias lógicas y realizar consultas interactivas, lo que facilita el desarrollo, prueba y análisis del simulador.

## Diseño de la solución

Para esta solución se ha definido la siguiente estructura de TDAs:

Sistema (system): Incluye las funciones propias al sistema de archivos que soportan las demás operaciones.

Unidad (Drive): Realiza las operaciones donde se implican discos o unidades.

Usuario (user): Contiene las funciones de tratamiento de usuarios como registro, login, logout.

Archivo (file): Contiene la creación del archivo y las propiedades y operaciones que se realizan sobre estos.

|  |
| --- |
| TDAs del Sistema |
| System  Drive  User  FilesSystem |

## Aspectos de implementación

El proyecto está diseñado lógicamente en 1 módulo que contiene las funciones necesarias para cumplir con el objetivo del proyecto.

“Físicamente” está estructurado en 7 archivos de código fuente Scheme con extensión .rkt:

1. main\_157809385\_HenriquezGonzalez.pl

Para el desarrollo de este proyecto e ha utilizado el IDE Web SWISH version V1.7.0-118-g8ece8ac-DIRTYRunning on SWI-Prolog version 9.1.8-18-ga51163963.

Sitio oficial (https://swish.swi-prolog.org/

## Instrucciones de uso

Las pruebas se deben realizar ejecutando el archivo:

*main\_157809385\_HenriquezGonzalez.pl*

Ejecutándose en el IDE Web SWISH

Desde la consola de consultas se puede copiar y pegar los códigos de pruebas incluidos en el archivo único del proyecto.

Ejemplo de ejecución:

|  |
| --- |
|  |
| Imagen 1: Consola de ejecución SWI Prolog |

## Resultados y autoevaluación:

## .

|  |
| --- |
| Autoevaluación por Requerimiento Funcional: |
| 0: No realizado. |
| 0.25: Implementación con problemas mayores (funciona 25% de las veces o no funciona) |
| 0.5: Implementación con funcionamiento irregular (funciona 50% de las veces) |
| 0.75: Implementación con problemas menores (funciona 75% de las veces) |
| 1: Implementación completa sin problemas (funciona 100% de las veces) |
| RF 1 = 1 System |
| RF 2 = 1 systemAddDrive |
| RF 3 = 1 systemRegister |
| RF 4 = 1 systemLogin |
| RF 5 = 1 systemLogout |
| RF 6 = 1 systemSwitchDrive |
| RF 7 = 1 systemMkdir |
| RF 8 = 1 systemCd |
| RF 9 = 1 systemAddFile |
| RF 10 = 0.75 systemDel |
| RF 11 = 0.75 systemCopy |
| RF 12 = 0.75 systemMove |
| RF 13 = 0.75 systemRen |
| RF 14 = 1 systemDir |
| RF 15 = 1 systemFormat |
| RF 16 = 0 systemEncrypt |
| RF 17 = 0 systemDecrypt |
| RF 18 = 0.75 systemGrep |
| RF 19 = 0.75 systemViewTrash |
| RF 20 = 0.75 systemRestore |

## Conclusiones del trabajo

## En conclusión, la construcción de un simulador de administrador de archivos en Prolog, desde el enfoque del paradigma lógico, presenta ventajas significativas. El paradigma lógico proporciona una base sólida para representar y modelar la estructura jerárquica de directorios y archivos, así como las operaciones y consultas asociadas.

## Desde un punto de vista académico y de aprendizaje, la construcción de un simulador de administrador de archivos en Prolog ha aportado varias conclusiones valiosas. En primer lugar, ha permitido comprender y aplicar los conceptos fundamentales del paradigma lógico, como la lógica de predicados y la inferencia lógica, en un contexto práctico. Esto ha fortalecido la comprensión de la programación lógica y su capacidad para abordar problemas complejos de manera declarativa.

## Además, este trabajo ha demostrado la utilidad de Prolog como herramienta para modelar sistemas complejos y simular su comportamiento. El uso de Prolog ha permitido representar de manera elegante y concisa la estructura de directorios y archivos, así como implementar las operaciones correspondientes. Esto ha resaltado la capacidad de Prolog para realizar inferencias y deducciones automáticas, lo que resulta valioso en el análisis y resolución de problemas del mundo real.

## Referencias

* SWI Prolog Reference Manual (n.d.). *SWI-Prolog documentation*. Retrieved May 15, 2023, from https://www.swi-prolog.org/pldoc/index.html
* Clocksin y Mellish. (2012). *Prolog Programming for Artificial Intelligence*. Página 16.