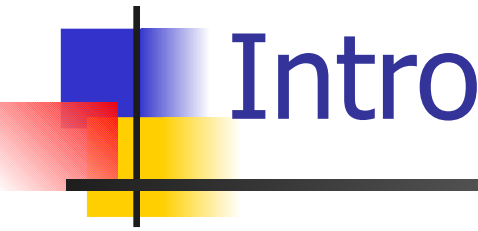


Introducción a SWI-Prolog

ETSI Informática
Dpto. Lenguajes y Ciencias
de la Computación
Universidad de Málaga

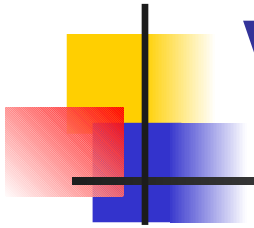
El GUI de SWI-Prolog

- Está basado en dos ventanas:
  - La **ventana principal**, con una línea donde se ejecutan los objetivos
  - La **ventana de edición**, donde se editan y compilan los programas
  - La ventana principal siempre existe, la de edición sólo cuando se está usando

Navegación por directorios

- Podemos usar tres predicados basados en Unix:
  - pwd
  - ls
  - cd

Ver directorio actual



- Predicado pwd/0
- Ejemplo:

```
?- pwd.
c:/src/prolog
```

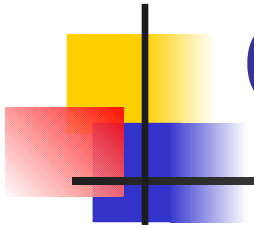
- Utiliza notación **Unix** para los directorios (‘/’ en lugar de ‘\’)

Ver contenido del directorio

- Predicado ls/0
- Ejemplo:

```
?- ls.
ejemplo.pl   principios.pl
intro.pl     proyecto/
```

Cambiar de directorio



- Predicado cd/1
- Ejemplo:

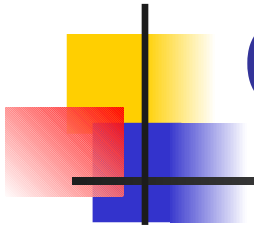
```
?- cd('e:/trabajo/prolog').
```

- La ruta aparece entre apóstrofos
- La ruta puede ser relativa o absoluta
- Recuerda: notación **Unix**

Edición de programas Prolog

- El predicado edit/1 permite:
  - Crear un programa nuevo
  - Editar un programa existente
  - Editar un predicado concreto
- La ventana de edición aparece al invocar a edit/1

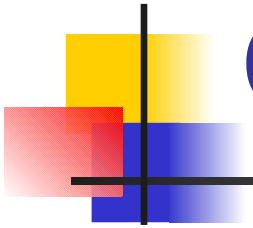
Crear un programa nuevo



- Ejemplo:
  `?- edit(file('ejemplo.pl')).`
- El functor `file` es imprescindible
- La extensión debe ser .pl
- Ruta: **Unix**, entre apóstrofos
- El fichero no debe existir
- El fichero creado será un fichero de texto **Unix** (aunque uses Windows)

Edita y guarda este programa

```
% quiere/2
quiere(juan, elena).
quiere(manuel, elena).
% celoso/2
celoso(X,Y) :-
        quiere(X,T),
        quier(Y,Z),
        X \== Y.
```



Coloreado del código

- El editor colorea el programa conforme lo editamos
- El coloreado atiende a razones sintácticas
- Puede ayudar a descubrir errores

Nuestro programa coloreado

```
% quiere/2
quiere(juan, elena).
quiere(manuel, elena).
% celoso/2
celoso(X,Y) :-
        quiere(X,T),
        quier(Y,Z),
        X \== Y.
```

Código de colores

- Predicado llamado: **`quiere`**
- Predicado no llamado: **`celoso`**
- Predicado indefinido: `quier`
- Predicado predefinido: `\==`
- Variable: `X`
- Variable unitaria (singleton): **`T`**
- Comentario: `% quiere/2`

Nuestro programa corregido

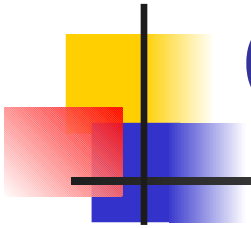
```
% quiere/2
quiere(juan, elena).
quiere(manuel, elena).
% celoso/2
celoso(X,Y) :-
        quiere(X,Z),
        quiere(Y,Z),
        X \== Y.
```

Otros elementos de resaltado

- Las llamadas recursivas se subrayan:

```
antepasado(X,Y) :-
        progenitor(X,Y).
antepasado(X,Y) :-
        progenitor(X,Z),
        antepasado(Z,Y).
```

- Al colocarse sobre una variable se destacan todas sus apariciones
- Al colocarse junto a un paréntesis se destaca su pareja



Compilar

- En el menú del editor, selecciona **Compile/Compile Buffer**
- Si hay errores aparecen en una ventana emergente
- Si el programa ha sido modificado se ofrece la posibilidad de guardarlo antes de compilarlo

Ejecutar

- Objetivo en la ventana principal
- Solicita más respuestas con ;

```
?- celoso(A,B).
A = juan
B = manuel ;
A = manuel
B = juan ;
No
```

Salir del entorno

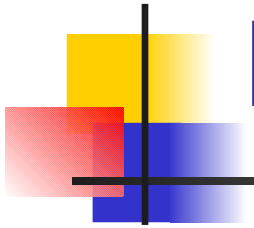
- Predicado halt/0

```
?- halt.
```

Editar un programa existente

- Ejemplo:
  ```
  ?- edit('ejemplo.pl').
  ```
- El fichero debe existir
- Se puede simplificar a:
  ```
  ?- edit(ejemplo).
  ```
  si 'ejemplo.pl' está en el directorio actual

Editar un predicado concreto

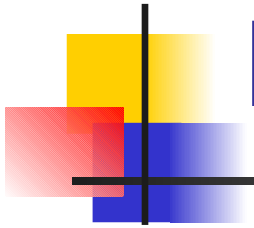


- Ejemplo:
  ```
  ?- edit(celoso/2).
  ```
- Hay que indicar nombre y aridad
- Sólo funciona si el predicado ha sido **compilado** previamente

Ayuda de SWI-Prolog

- Predicado help/0
  Abre la ventana de ayuda navegable:
  `?- help.`
- Predicado help/1
  Abre la venta de ayuda y muestra información sobre el argumento:
  `?- help(edit/1).`

Edición de la línea de órdenes



- Historial: cursor arriba/abajo
- Compleción automática: TAB

```
?- cel<TAB>
```

- Corrección (Do What I Mean):

```
?- celos(A,B).
Correct to: celoso(A, B)?
```