# Sintaxis de configuración

## Sintaxis básica

Como es tradicional en los ficheros INI, toda la información de configuración es expresada usando parejas de llave/valor.
Estas parejas están organizadas en secciones.

Una sección es declarada con corchetes ([] - ES-ES). Todas las parejas de llave/valor definidas debajo serán parte de esta sección.
Esta sección termina cuando otra comienza.

```
[Section]
```

Llaves y valores son delimitados con signo de igual (=).

```
Key = Value
```

Punto y coma (;) indica el comienzo de un comentario. El comentario continua hasta el final de la linea.

```
; Esto es un comentario y será ignorado por el módulo de configuración
```

Las secciones pueden ser definidas en más de un lugar. Incluso pueden abarcar varios ficheros de configuración.

He aquí un ejemplo.

```
; Aquí tenemos un ejemplo de una sintaxis de configuración de orx
[MySection] ; Esto define el inicio de 'MiSección'('MySection')
MyKey      = MyValue; Aquñi damos un valor a 'MiLlave'('MyKey')
MyOtherKey = MyOtherValue; Y aquí damos uno para 'MiOtraLlave('MyOtherKey')

[MyOtherSection]; Aquí termina 'MySection' y estamos ahora en
'MiOtraSección'('MyOtherSection')
AKey = Otro valor

[MySection]; Estamos de regreso a 'MySection'
MyLastKey = MyLastValue; Añadiendo otra pareja llave/valor a 'MySection'
```

**PD: Los espacios alrededor del operador de asignación ('=') son recortados y simplemente ignorados por la configuración del sistema.**

Si quieres usar un ';' como parte de un valor no-númerico, necesitas usar una asignación en bloque. Los bloques son delimitados con comillas dobles . Los bloques también son la única manera de tener valores cubriendo varias lineas.

```
MyKey      = "MyValuePart1 ; MyValuePart2"
```

```
MyOtherKey = "Este valor
se extiende
en múltiples lineas"
```

Si doblas las primeras , la cadena no será considerada como un bloque pero si como un valor normal y tendremos un como inicio del valor.

```
MyKey = ""MyQuotedValue"
```

Aquí la cadena MiValorCitado(MyQuotedValued) (incluyendo las comillas dobles), serán almacenadas como un valor para la llave 'MyKey'.

## Herencia

The inheritance system is based on the same idea than inheritance in object-oriented programming.

The basic idea is that all the keys defined in a section can be inherited by any other section [1].
The inheriting section (aka the child section) can then add new keys or even override any key defined in the parent section.

In order to do so, the arobase ('@') is used as an inheritance marker.

```
[Parent]
MyKey1 = MyValue1
MyKey2 = MyValue2

[Child@Parent]; <= The 'Child' section now contains all key/value pairs
defined in the 'Parent' section
```

If you don't want to inherit a whole section, you can also use inheritance directly for a single key.
If the parent key doesn't have the same name as the child one, you can specify its complete name using the dot ('.') separator in addition to the inheritance marker.

```
[Parent]
MyKey      = MyValue
MyOtherKey = MyOtherValue

[Child]
MyKey     = @Parent; <= The value for 'MyKey' will be inherited from the one
defined in the 'Parent' section, with the same key name.
MyLastKey = @Parent.MyKey; <= The value for 'MyLastKey' will also be
inherited from the key 'MyKey' defined in the 'Parent' section.
```

In the previous example, we see that both 'MyKey' and 'MyLastKey' of the 'Child' section inherit from 'MyKey' of the 'Parent' section.
In this example, the key 'MyOtherKey' won't be inherited in the section 'Child'.

When using inheritance, when the parent key's value change, even at runtime, the child value will

also be changed.
Inherited values can be chained as seen in the example below.

```
[GrandParent]
MyKey      = MyValue
MyOtherKey = MyOtherValue

[Parent]
MyKey = @GrandParent

[Child@Parent]
```

In the section 'Child', there's only one key defined called 'MyKey'. Its value is inherited from the 'Parent' section who already inherits it from the 'GrandParent' one. In other words, when `GrandParent.MyKey` changes, both `Parent.MyKey` and `Child.MyKey` will be affected.

# Includes

# Valores numéricos

### Tipos básicos

### Vector

### Aleatorio

# Listas

[1)]
cyclic inheritance is **not** supported