

## Introduction to (Extended) Backus-Naur Form, (E)BNF

The Extended Backus-Naur Formalism (BNF) notation makes use of several special symbols. These are listed below, together with their meanings:

### Symbol Usage

> <	Enclose term names
	Separates alternatives (exclusive OR)
[]	Term enclosed is optional (not used or used once)
{ }	Term enclosed is used zero or more times
()	Enclose groups of alternative terms

### BNF notation for syntax

#### **(a) Notational Conventions**

This specification uses an augmented Backus-Naur Form (BNF) notation. The differences from standard BNF involve naming rules and indicating repetition and "local" alternatives.

##### 1. Rule Naming

Angle brackets "<" and ">" are not used, in general.

The name of a rule is simply the **name** itself, rather than <name>

Quotation-marks enclose literal text (which may be upper and/or lower case). Certain basic rules are in uppercase, such as SPACE, TAB, CRLF, DIGIT, ALPHA. Angle brackets are used in rule definitions.

##### 2. Rule1 / Rule2: Alternatives

Elements separated by slash ("/") are alternatives. Therefore "foo / bar" will accept foo or bar.

NOTE: this rule is changed to use the vertical bar character "|" instead of slash, since the syntax for directory paths uses slashes heavily, i.e. we now use "foo | bar".

##### 3. (Rule1 Rule2): local Alternatives

Elements enclosed in parentheses are treated as a single element. Thus, "(elem (foo | bar) elem)" allows the token sequences "elem foo elem" and "elem bar elem".

##### 4. \*Rule: Repetition

The character "\*" preceding an element indicates repetition. The full form is:

<l>\*<m>element

indicating at least l and at most m occurrences of element. Default values are 0 and infinity so that

"\*(element)" allows **any number, including zero**; "1\*element" requires at least one; and

"1\*2element" allows one or two.

##### 5. [Rule]: Optional

Square brackets enclose optional elements; "[foo bar]" is equivalent to "\*1(foo bar)".

##### 6. NRule: Specific Repetition

`"<n>(element) "`  
is equivalent to  
`"<n>*<n>(element) "`

that is, **exactly n occurrences of (element)**. Thus 2DIGIT is a 2-digit number, and 3ALPHA is a string of three alphabetic characters.

### 7. #Rule: Lists

A construct "#" is defined, similar to "\*", as follows:

`<l>#<m>element`

indicating **at least l and at most m elements, each separated by one or more commas (",")**. This makes the usual form of lists very easy; a rule such as `'(element * (" , " element))'` can be shown as `"1#element"`. Wherever this construct is used, null elements are allowed, but do not contribute to the count of elements present. That is, `"(element) , , (element)"` is permitted, but counts as only two elements. Therefore, where at least one element is required, at least one non-null element must be present. Default values are 0 and infinity so that `"#(element) "` allows any number, including zero; `"1#element"` requires at least one; and `"1#2element"` allows one or two.

### 8. ; Comments

A semi-colon, set off some distance to the right of rule text, starts a comment that continues to the end of line. This is a simple way of including useful notes in parallel with the specifications.

### 9 A rule is written as

`NonTerminal ::= RuleOfTerminalsAndNonTerminals`

### **(b) General rules**

Sometimes it is necessary to specify a layout character sequence, such as `newline`. Sometimes the convention used is as in C strings, sometimes abbreviations are used:

<code>\n</code>	LF	newline
<code>\r</code>	CR	carriage-return
<code>\t</code>	TAB	tab
<code>\b</code>	BS	backspace
<code>\f</code>	FF	orm feed

### Strings (i.e. text in double quotes):

The purpose of a string is to show mandatory items. A string is a sequence of ASCII characters. Thus, a string is surrounded by double-quote characters and excludes control characters. If a double-quote character is used inside a string, it must appear as the sequence `\"` (backslash followed by double quote).

A control character can be represented by a backslash followed by its ASCII sequence number in octal notation. The NUL is not representable.