

## ● Builtin PREDICATION

Unlike a library module, an builtin predicate is called without specifying a library-module name.

---

<alt PRED...>

Alternative predicate execution is shown.  
Even if the results of Argument PRED are true, false, and unknown, it is certainly set to true.  
It expresses [PRED...] as substitution. It uses by syntax analysis as [] of the method of EBNF.

<assert CLAUSE>  
<asserta CLAUSE>  
<assertz CLAUSE>

CLAUSE is added.  
asserta is added to a head and assertz is added at the end.  
assert is the same as asserta and is added to a head.

<cd PATH>

A current directory is moved to the PATH specified by the argument.

<edit FILE-NAME>

A FILE-NAME is edited using the editor set as the environment variables DEDITORPATH and EDITOR.

<dir>  
<ls>

A current directory is displayed.

<compare EXPRESSION>  
<comparef EXPRESSION>

EXPRESSION is evaluated.  
compare is an integer and comparef is comparison of a float number.

<erase CLAUSE-NAME>  
<retract CLAUSE-NAME>

All the programs corresponding to CLAUSE-NAME are deleted.

<retractpred HEAD-NAME>

All the programs corresponding to HEAD-NAME are deleted.

<f VAR PRED...>  
<func VAR PRED...>

Execution evaluation of the function predicate contained in PRED-LIST is carried out. A value is replaced. After performing all, a result is set to a variable.

<findall PRED...>

PRED is performed and all the solutions are calculated.

<for (VAR lastvalue) PRED...>  
<for (VAR initialvalue lastvalue) PRED...>

The predicate of the specified number of times and an argument is performed.  
When the initial value is not specified, the value from 0 is set to a VAR.

<foreach (VAR LIST) PRED...>  
<map (VAR LIST) PRED...>

PRED of arguments are performed for every element of LIST.  
The value of a list is set to a variable in order.  
<include FILENAME>  
The library of a FILE-NAME is read.

<let EXPRESSION>  
<letf EXPRESSION>

Expression is calculated.  
A calculation result is substituted when the left side is a variable.  
When the left side is a numerical value, it is judged whether it is equal to a calculation result.  
let calculates an integer and letf is calculation of floating number.

<list>

A program list is displayed.

<load FILE-NAME>

The program of a FILE-NAME is read.

<loop PRED...>

A repetition of predicate execution is shown.  
It was called inside. Failure of a predicate will escape from a loop.  
It is the predicate which is surely successful by true,  
It expresses {PRED...} as substitution.  
It uses by syntax analysis as {} of the method of EBNF.

<module VAR>

The library module used now is set to a variable.

<new>

All log rum is cleared.

<not PRED...>

false is returned when predicate execution is true.

In false, true is returned.

In unknown, unknown is returned.

<obj OBJECTNAME PRED...>

<unify MODULENAME PRED...>

A predicate is performed using the specified object or module.

A module is the same as an object.

It can be described as "::object <PRED...> as an omitted type.

<or PRED PRED PRED ...>

In the place which performed the predicate of the argument from the head and was set to true Processing is closed and true is returned.

In the place which performed the predicate of the argument from the head and was set to false Processing is closed and false is returned.

unknown is returned when all the predicates are unknown.

It uses by syntax analysis as | of the method of EBNF.

<print LIST>

fter outputting a list, a new line is started.

<writeln LIST> is the same as <print LIST>.

<pwd VAR>

The current directory is set to VAR.

<quit>

Execution of a program is stopped and it ends.

<quote PRED>

Evaluation of the function predicate of an argument is deterred.

<rpn VAR RPNEXPRESSION>

<rpnf VAR RPNEXPRESSION>

A reverse Poland style is calculated and a result is set to a variable.

rpn calculates an integer and rpnf calculates floating number.

<save FILE-NAME>

The program of a FILE-NAME is written in.

<timeout limittime PRED>

Within the set-up time, when execution of a predicate is not completed, processing is closed, and unknown is returned. A set period is a micro second bit.

<troff>

Debugging trace is turned OFF.

<tron>

Debugging trace is turned ON.

<true>

<false>

<unknown>

return true, false, unknown

<!>

cut operator

EBNF Syntax analysis

<TOKEN VAR PRED...>

After syntax-analysis PRED execution of an input, obtained token is set as VAR.

<SKIPSPACE>

The space of an input is skipped.

<C [VAR]>

An input is set as one-character VAR.

<N [VAR]>

When an input is a number, it is set as VAR.  
unknown is returned when different.

<A [VAR]>

When an input is the ASCII character, it is set as VAR.  
unknown is returned when different.

<AN [VAR]>

When an input is the ASCII character or a number,  
it is set as VAR. unknown is returned when different.

<^>

The head of a line is matched.

<\$>

The last of a line is matched.

<\* VAR>

Arbitrary character strings are matched.

<CR [VAR]>

When an input is CR new-line, it is set as VAR.  
unknown is returned when different.

<CNTL [VAR]>

When an input is the CNTL character, it is set as VAR.  
unknown is returned when different.

<EOF [VAR]>

When an input is the EOF character, it is set as VAR.  
unknown is returned when different.

<SPACE>

true is returned when an input is a space. unknown is  
returned when different.

<PUNCT>

true is returned when it is characters other than the  
alphabet and a number.

<STRINGS VAR>

STRINGS of "... " or '...' is matched, and it is set  
as VAR.

<WORD VAR>

In STRINGS(s) other than the alphabet, a number,  
and "\_", unknown is returned by arbitrary STRINGS.

<NUM VAR>

The integer of an input is set as VAR.

<FNUM VAR>

The floating point number of an input is set as VAR.

<ID VAR>

If an input STRINGS (a head is the alphabet and a number is also good except it) and it agrees, it will be set as VAR.

<RANGE VAR CHAR1 CHAR2>  
<NONRANGE VAR CHAR1 CHAR2>

It will be set to true if contained in the range of a character 1 and a character 2.

<GETTOKEN VAR>

The token which is a result of the last syntax analysis is set as VAR.