● sys モジュール述語

sysモジュールはシステムに組み込まれた標準のライブラリ
モジュールの集合です。
呼び出すときには、"::sys"に続けて記述します。

例)
::sys <args #x>;

<args 変数>

変数に、デカルト言語を起動したときの引数を設定
します。

<DLIBPATH 変数>

デカルト言語の使用するパスDLIBPATHを変数に表示
します。

<cutall>

ここまでの実行履歴をすべてカットします。

<mkpred 引数>

引数を述語に変換します。

<writenl リスト>

リストを出力した後、改行します。

<write リスト>
<w リスト>

リストを出力します。

<wnl>

改行を出力します。

<wo 数値>

数値を８進数で出力します。

<wx 数値>

数値を16進数で出力します。

<wf 数値>

数値を浮動小数点数として出力します。

<wg 数値>

数値を浮動小数点数と最適な形式で出力します。

<wtab>

タブを出力します。

<fr 変数 文字列 幅>

文字列を幅で右寄せした結果を変数に設定します。

<fl 変数 文字列 幅>

文字列を幅で左寄せした結果を変数に設定します。

<isNil 引数>
<isAtom 引数>
<isList 引数>
<isPred 引数>
<isVar 引数>
<isUndefVar 引数>
<isFloat 引数>
<isInteger 引数>

引数を判定して該当すればtrueを返します。該当
しなければunknownを返します。

<isTrue 述語>
<isFalse 述語>
<isUnknown 述語>

引数の述語の結果を判定し、該当すればtrueを返します。
該当しなければunknownを返します。

<regex 正規表現パターン 文字列 前文字列 マッチ文字列 後文字列>

文字列に正規表現パターンを適用した結果をマッチ文字列
に設定し、前後の文字列を前文字列と後文字列に設定します。
(Windows上では動作しません。)

<sub 正規表現パターン 文字列 置換文字列 出力文字列>

文字列に正規表現パターンを適用して該当した部分を、
置換文字列に置換えた結果を出力文字列に設定します。
置換えは最初の１回だけ行われます。
(Windows上では動作しません。)

<gsub 正規表現パターン 文字列 置換文字列 出力文字列>

文字列に正規表現パターンを適用して該当した部分を、
置換文字列に置換えた結果を出力文字列に設定します。
置換えは文字列のすべての該当部分に行われます。
(Windows上では動作しません。)

<split 変数 文字列 [区切り文字]>

文字列を区切り文字で分けてリストにしたものを

変数に設定します。区切り文字が指定されていない
場合は、空白とタブで区切られます。

〈length 変数 リスト〉

リストの長さを変数に設定します。

〈setvar 変数名 値〉

グローバル変数に値を設定します。
グローバル変数は、述語として以下の形式で登録
されます。
〈変数名 値〉;

〈setarray 変数名 値 インデックス〉

グローバル変数配列に値を設定します。
グローバル変数は、述語として以下の形式で登録
されます。
〈変数名 値 インデックス〉;
多次元の配列を仕様する場合は、インデックスを
リストとして指定します。
〈変数名 値 (インデックス1 インデックス2 ...)〉
インデックスは数以外にもいかなる種類や形式で
も良いです。

〈random 変数〉

変数に乱数を設定します。

〈sin 変数 ラジアン〉
〈cos 変数 ラジアン〉
〈tan 変数 ラジアン〉

三角関数

〈asin 変数 値〉
〈acos 変数 値〉
〈atan 変数 値〉

逆三角関数

〈log 変数 値〉
〈exp 変数 値〉

対数関数

〈sqrt 変数 値〉

平方根

〈abs 変数 値〉

絶対値

〈int 変数 値〉

整数値

<car 変数 値>
<cdr 変数 値>

リストのcar, cdr

<cons 変数 リスト1 リスト2>

リストの連結

<code コード>

文字コードの設定。UTF8, EUCJP, SJISが指定できます。

<char 変数 文字列>

文字列を文字ごとに分解してリストにして、変数に設定します。漢字のような多バイト文字も正しく一文字ごとに分解します。

<byte 変数 文字列>

文字列をバイトごとに分解してリストにして変数に設定します。漢字のような多バイト文字もバイト単位に分解されます。

<asciichar 変数 文字列>
<utf8char 変数 文字列>
<eucchar 変数 文字列>
<sjischar 変数 文字列>

文字列を文字コードごとに分解してリストにして変数に設定します。

<concat 変数 リスト>

リストを合体させ文字列を合成して変数に設定します。

<bitand 変数 数値1 数値2>
<bitor 変数 数値1 数値2>
<bitxor 変数 数値1 数値2>
<bitnot 変数 数値1>

bit演算

<shiftl 変数 数値 シフト数>
<shiftr 変数 数値 シフト数>

整数値のビットシフト。
shiftlは左へ、shifutrは右へシフトさせます。

<eq 引数1 引数2>
<noteq 引数1 引数2>
<is 引数1 引数2>

引数1と引数2の比較

<getc 変数>

変数に１char入力します。

<putc 文字>

1charを出力します。

<getline 変数 [述語...]>

1行入力して変数に設定します。述語が設定されている場合は
tmpファイルを入力として述語が実行されます。

<tmpfile 変数>

テンポラリファイル名を変数に設定します。

EBNF 構文解析

<TOKEN 変数 述語...>

入力の構文解析述語実行後に、得られたtokenを変数に
設定します。

<SKIPSPACE>

入力のスペースをスキップします。

<C [変数]>

入力を一文字変数に設定します。

<N [変数]>

入力が数字であった場合は、変数に設定します。
違う場合はunknownを返します。

<A [変数]>

入力がASCII文字であった場合は、変数に設定します。
違う場合はunknownを返します。

<AN [変数]>

入力がASCII文字か数字であった場合は、変数に
設定します。
違う場合はunknownを返します。

<CR [変数]>

入力がCR改行であった場合には、変数に設定します。
違う場合はunknownを返します。

〈CNTL [変数]〉

入力がCNTL文字であった場合には、変数に設定します。
違う場合はunknownを返します。

〈EOF [変数]〉

入力がEOF文字であった場合には、変数に設定します。
違う場合はunknownを返します。

〈SPACE〉

入力がスペースである場合はtrueを返します。
違う場合はunknownを返します。

〈PUNCT〉

アルファベット、数字以外の文字である場合はtrue
を返します。

〈STRINGS 変数〉

"～"または'～'の文字列とマッチして、変数に設定
します。

〈WORD 変数〉

任意の文字列で、アルファベット、数字、"_"以外
の文字列の場合はunknownを返します。

〈NUM 変数〉

入力の整数を変換して、変数に設定します。

〈FNUM 変数〉

入力の浮動小数点数を変換して、変数に設定します。

〈ID 変数〉

入力の文字列（先頭はアルファベット、それ以外
は数字も可)、合致すれば変数に設定します。）

〈RANGE 変数 文字１ 文字２〉
〈NONRANGE 変数 文字１ 文字２〉

文字１と文字２の範囲に含まれるならばtrue
となります。

〈GETTOKEN 変数〉

直前の構文解析の結果であるトークンを変数に設定
します。

〈openr ファイル名 述語...〉

ファイル名のファイルを読み取り用にオープンして、
述語を実行します。

〈openw ファイル名 述語...〉

ファイル名のファイルを書き込み用にオープンして、
述語を実行します。

〈openwp ファイル名 述語...〉

ファイル名のファイルを追記書き込み用にオープン
して、述語を実行します。

〈countnode 変数〉

使用しているノード数を変数に設定します。

〈gc〉

ガーベージコレクタを起動します。

以上。