note-stream 2019-11-26 11:27:57

[R5RS] stream

(input-port? obj)

(output-port? obj)

(current-input-port)

(current-output-port)

(close-input-port port)

(close-output-port port)

(open-input-file filename)

(open-output-file filename) [R5RS,+]
　　すでに filename が存在する場合それは上書きされます。

(call-with-input-file file proc)

(call-with-output-file file proc)

(char-ready? [port])
　　port が入力可能なら#t を返す。port が socket の場合は
　　可能な入力バイト数または#t を返す。
　　入力がなければ#f を返す。

(read [port])

(read-char [port])
　漢字モード (EUC) のときは euc コードは 2byte を 1 個の文字として read
　する。

(peek-char [port])
　　This is almost same as
　　(unread-char(read-char port) port)

(eof-object? obj)

(write obj [port])

(display obj [port])

(newline [port])

(write-char char [port])

[SS]
　(read-line [port])
　　改行が現われるまで読みこんだ文字列を返す。改行コードは
　　文字列に含まれない。SJIS の改行 0d0a, MAC の改行 0a は 1 個の
　　改行コードとして処理する。

　(read-byte-char [port])
　　漢字モードに関係なく 1byte ずつ read する。

　(read-number [path])
　結果は常に実数または eof-obj である
　高速性が要求される場合に１０進実数
　の read のみに使う
　空白と tab, 改行は無視する。

[SS] port stream
　ss で扱えるストリームは、標準入出力、ファイル、socket, 文字列、関数型
　である。すべてのストリームに対して、read,write,format,load などの
　関数を使用できる。

　(close-port port)

　(closed-port? obj)

　(socket-port? obj)

　(current-error-output-port)

　(flush [output-port])

　(file-exists? filename)

　(open-append-file filename)
　　追加書き込みで open される。

　(open-input-str str)

(open-output-str str)

(open-append-str str)

(open-io-str str)
(open-ia-str str)

これらは、文字列に対するポートを作る。書き込みの場合
もとの文字列の大きさは自動的に拡大される。
結果の文字列は、これを close したときの返り値である。
open-io-str では str は input,output モードで open される。

open-ia-str では str は input,append モードで open される。

(reopen-str str-port [:input|:ouput|:append|
:io|:ia])
str-port はすでに close された文字列ポートで、それが
作られたときと同じモードで open される。
mode を指定してもよい。

(open-input-func funcion [p1 ...])
関数型の入力ポートを作る。
関数は 2 個の引数を受けつけ、以下のように動作しなければならない。
弟 2 引数で与えられるのは通常 eof-object である
弟 1 引数の値が:open のときは初期化を行なう。これは、open 処理のとき
の呼びだしである。初期化に失敗したときは関数内部で error を
起こすか、#f を返す。もし、unread-char をこの関数で独自に監理するときは
:open に対する帰り値はキーワード :unread でなければならない。
:open の場合の弟 2 引数は、p1 ... のリストである。
弟 1 引数の値が:close のときは close 処理を行なう。
弟 1 引数の値が:ready のときは文字が入力可能か end-of-file なら#t
そうでなければ
#f を返す。。
弟 1 引数の値が:input のときは 1 個の文字または eof-object を返す.
弟 1 引数の値が :unread のとき第 2 引数は文字であり、関数はその文字を
値として返す。
弟 1 引数の値が :reopen のとき弟 2 引数は reopen-func に与えられたすべての
引数のリストである。 :reopen をサポートする場合、帰り値は真である。
[read-pos (optional)]
:readpos see read-pos.

[set-read-pos (optional)]
:set_readpos see set-read-pos.

[shift-read-pos (optional)]
:shift_readpos see shift-read-pos.

弟 1 引数がこれらでないときは#f を返す。これは将来の拡張のためである。
(このときの弟 2 引数は eof-object でないかもしれない。)

unread-char の処理は関数が処理しないのならシステムが行なう。
日本語処理は system が行なう。(関数において行なっても良い)

(open-output-func funcion [p1 ...])
関数型の出力ポートを作る
関数は 2 個の引数を受けつけ、以下のように動作しなければならない。
弟 2 引数で与えられるのは通常出力するべき文字である。
弟 1 引数の:open,:close の意味は前と同様である。
close-port 関数は、この関数が close 処理で返す値を返す。
弟 1 引数の値が:flush のときは、関数は flush 動作を行なう。
flush が返す値も、このときの関数が返す値である。
弟 1 引数の値が :output のときは弟 2 引数で与えられた
文字を出力する。
:unwrite The second argument is eof-ob
ject. See unwrite
:getpos See get-line-pos
:setpos See set-line-pos

:unread,:unwrite,:getpos,:setpos,:readpos
,:set_readpos, :shift_readpos の
いくつかを サポートするときは、 :open に対してそれらのキーワードの
リストを返さなければならない。

(open-io-func funcion [p1 ...])
入力出力の両方可能な関数型のポートを作る。

(reopen-func func-port [p1 ...])

(call-with-input-str str proc)

(call-with-output-str str proc)

(with-input-from-file file thunk) [R5RS,+]

(with-output-to-file file thunk) [R5RS,+]

(end-of-file? port)

(unread-char char [port])
入力 port に 1 文字戻す。これを連続して行なうことは通常 error である。 関数 port を見よ。

(unwrite-char port)
port は str または関数型である。書き込み位置を 1 つ前に戻す。
返される値は取り消された文字である。もし、位置がすでに先頭ならば、eof-object が返される。

(inkey [echo-flag] [wait-flag])
この関数を使用するためには scheme が起動したときの標準入力は端末でなければいけない。
これは端末から全く buffering なしで 1 文字入力する。
echo-flag,wait-flag の default は#t である。
(read-char) と (inkey) は似ているがかなり動作は異なる。
(read-char) は buffer 動作を行なうので。1 文字の入力だけでは終了しない。

(copy-port in out)
入力ポート in の内容をすべて out に書き出す。

(file->string file)

(port->string port)

(do-read proc)
proc は 1 引数の関数 object である。標準入力から read した object に対し proc を適用する。これを入力が eof になるまで行なう。

(do-read-line proc)

(port-filename port)
port が file に対するものなら、その file 名を返す。それ以外は#f.

(port-string port)

(port-function port)

(fileno port)
port が Unix の fileno(整数値) を持つならそれを返す。そうでなければ#f を返す。

(stdio-push! integer port)
integer は 0,1,2 のどれかである。integer=0 ならば (current-input-port) の値が port になるように変更される。
これは stack として動作する。
redirect が、fileno の操作を伴うのに対しこれは単に scheme の内部における標準 port を変更する。

(stdio-pull! integer)
もとにもどす。close-port は別にする必要がある。

(read-pos port)
(set-read-pos num port)
(shift-read-pos num port)
port must be a string or function port.
These handle read position.
Do not use these functions with unread-char or peek-char.
It may cause confusion.

(get-line-pos [output-port])
行出力の位置 (行頭を 0 とする。) を整数値で返す。
この関数は制御コードなどは認識しない。
最後に改行が出力されてから何文字が出力されたかを示す。

(set-line-pos pos [output-port])

(load filename-or-port) [R5RS,+]

(load-verbose [read-echo value-echo search-e

cho])

load 関数の情報表示を制御する。3 個の引数の default は真である。

read-echo は、ファイルから read した行を表示する。

value-echo は、eval の結果を表示。

search-echo は、load 関数が search する path 名を表示する。

compile された*.sr ファイルに対しては read-echo,value-echo は
無効である。
引数のないときは、それらの値のリストを返す。

stdnull 出力ポートだが、このポートに対するすべての出力は単に
捨てられる。

[SS] 起動 初期化ファイル

ss が起動したとき

initss.ss (または initsn.ss for sn)

をさがす。 search は一般に

./

の次に*auto-path*(これは directory のリストである。) を見てその先頭から順に

行なわれる。*auto-path*の初期値は

("/usr/local/lib/sslib/")

というリストである。必要ならこれを変更することが許されている。dir 名の最後に/が必要なことに注意してほしい。

変数*SSLIBDIR*には 文字列"/usr/local/lib/sslib/"が set されている。

initss.ss を load したあと

./start.sr (./start.ss)

があれば load されて、最後に起動コマンドのパラメータで渡された
file が load されて start する

拡張子が省略されたときは，拡張子なしを最初に見てから

.sr, .ss, .scm, .lsp,.stk

の順に search が行われる。.sr は compile された file である。/usr/local/lib/sslib/を変更するには、Makefile を書き変えて make する必要がある。

initss.ss を他の初期化ファイルに指定したいときは

ss -i hoge.ss

のように -i オプションを使う。