myload.ss 2019-11-20 15:16:48

```
(shadowing-intern "load"  "sys")

(define (mload f)
  (let* ((org sys:*package*) (ret 0))
 ;; (format #t "mload store package ~s~%" org)
   (set! ret
     (cond ((string? f) (myload f))
         ((input-port? f) (load-port f))
         (else (error E_open f))
     )
   )
  (set! sys:*package*  org)
  ;;(format #t "mload restore package ~s~%" org)
  ret
))

(define (myload file)
 (let* ((ff 0)(ext 0))
   (set! ff (find-source file))
   (if (not ff) (error E_open file))
   (if (equal? ".sr2" (cdr (file-sepa-ext ff)))
       (loadsr2 ff)
       (load-source ff)
   )
))

(define (load-eval-error  e p)
    (when (not (eq? e E_hookreturn))
      (format *stdmsg* "load eval error~%")
      (print-error e p)
    )
)
(define (load-read-error e p)
      (print-error e p)
)

(define (read-safe in)
  (let* ((x 0))
    (set! x  (catch-error (lambda ()(read in)) load-read-error))
   x
))

(define (load-source file)
  (let* ((in (open-input-file file)) )
    (if in (begin (load-source-port in (read-safe in)) file)
        (error E_open file)
    )
))

(define (load-source-port in x)
  (let* ((ver (load-verbose)) (r-echo (pop ver))
          (v-echo (pop ver)) (z 0)(etop 0) (memo *errorhook*)
      (err #f)(change 0))
     (set!  z (call/cc (lambda(e) (set! etop e) etop)))
     (if (eq? z etop)
          (set! *errorhook* (lambda (e obj)(load-eval-error e obj) (etop 0 )))
         (begin
            (set! err #t)
            (format #t "load error hook return~%")
         ;;; (set! *errorhook* (lambda (e obj)(load-eval-error e obj) (etop 0 )))
            (set! x (read-safe in))
         )
    )
    (set! change *errorhook*)
    (while (not (eof-object? x))
      (if r-echo (format *stdmsg* "~s~%" x))
      (set! x (vm-compile x ()))
      (set! x  (CALL-VM x ))
      (if v-echo (format *stdmsg* "value: ~s~%" x)
      )
      (if (closed-port? in)
         (set! x *eof-object*)   ;;;  this need for call/cc , then port is closed
         (set! x (read-safe in))
      )
     )
    (if (eq? change *errorhook*)
       (if (not err)
          ;;; (set! *errorhook* memo)
       )
     )
    (close-port in)
 ))
```

```
(define  (load-port in)
  (let* ((x 0))
    (set! x  (read-safe in))
    (if (eq? x ’BEGIN-SR2)
       (begin (loadsr2-port in x) x)
       (load-source-port in x)
    )
))

(define  (loadsr2 file)
  (let* ((in (open-input-file file)) )
     (if in (begin (loadsr2-port in (read-safe in)
     ) file)
        (error E_open file)
     )
))

(define  (loadsr2-port  in x)
    ;;;(set! x  (read-safe in))
    (if (eq? x ’BEGIN-SR2)
      (let* ( (z 0)(etop 0) (memo *errorhook*)(err
       #f)(change 0))
        (set!  z (call/cc (lambda(e) (set! etop e)
         etop)))
         (if (eq? z etop)
            (set! *errorhook* (lambda (e obj)(load
            -eval-error e obj) (etop )))
            (begin
              (set! x (read-safe in))
     ;;;   (set! *errorhook* (lambda (e obj)
     (load-eval-error e obj) (etop )))
        (set! err #t)
      )
    )
  (set! x (read-safe in))
  (set! change *errorhook*)
;;; (format *stdmsg* "sr2-read: ~s~%" x)
  (while (not (eof-object? x))
    (set! x (CALL-VM x) )
;;; (format *stdmsg* "sr2-eval: ~s~%" x)
     (if (closed-port? in)
        (set! x *eof-object*)
        (set! x (read-safe in))
     )
  )
   (if (eq? change *errorhook*)
        (if (not err)
      ;;;;      (set! *errorhook* memo)
        )
   )
 )
 (format *stdmsg*   "Not sr2 port ~s~%" in)
)
(close-port in)
)

(set! load mload)
```