e-POSIX

**eposix short-flat
listing of classes**

written by Berend de Boer

Contents

A
Short (flat) listing of Standard C classes

A.1   Short form of STDC_BASE

class *interface STDC_BASE*
feature(*s*) **from** *STDC_BASE*
  -- Access
  errno: *STDC_ERRNO*
    -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
  -- Status
  raise_exception_on_error: *BOOLEAN*
    -- Should an exception be raised when an error occurs?
    -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
  -- Change
  set_default_action_on_error
    -- Use *security.error_handling.exceptions_enabled* to
    -- determine if an exception should be raised when a C call
    -- returns an error.
  set_raise_exception_on_error
    -- Always raise an exception when a C call returns an error.
  set_continue_on_error
    -- Never raise an exception when a C call returns an error.
  inherit_error_handling (*an_instance*: *STDC_BASE*)
    -- Handle errors like *an_instance*
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action >= 0* **and** *error_action <= 2*;
end *of STDC_BASE*

A.2 Short form of STDC_BUFFER

class *interface STDC_BUFFER*
creation
   allocate (*a_capacity*: *INTEGER*)
     -- Allocate memory of *a_capacity* bytes.
     -- If *is_owner* then the buffer is first deallocated.
   allocate_and_clear (*a_capacity*: *INTEGER*)
     -- Allocate memory of *a_capacity* bytes, make sure its zeroed out.
     -- If *is_owner* then the buffer is first deallocated.
   make_from_pointer (*a_pointer*: *POINTER*; *a_capacity*: *INTEGER*; *a_become_owner*:
BOOLEAN)
     -- Attach a pointer to this object. If *a_become_owner* is
     -- True, it will deallocate the pointer when *close* is
     -- called, or when this object is garbage collected.
feature(*s*) **from** *STDC_BUFFER*
  -- Allocation
   allocate (*a_capacity*: *INTEGER*)
     -- Allocate memory of *a_capacity* bytes.
     -- If *is_owner* then the buffer is first deallocated.
   allocate_and_clear (*a_capacity*: *INTEGER*)
     -- Allocate memory of *a_capacity* bytes, make sure its zeroed out.
     -- If *is_owner* then the buffer is first deallocated.
   make_from_pointer (*a_pointer*: *POINTER*; *a_capacity*: *INTEGER*; *a_become_owner*:
BOOLEAN)
     -- Attach a pointer to this object. If *a_become_owner* is
     -- True, it will deallocate the pointer when *close* is
     -- called, or when this object is garbage collected.
feature(*s*) **from** *STDC_BUFFER*
  -- Other allocation commands
   resize (*new_capacity*: *INTEGER*)
     -- Resize memory to *new_capacity* bytes. Expanded memory is not
     -- guaranteed to be zeroed out.
feature(*s*) **from** *STDC_BUFFER*
  -- Element change
   copy (*other*: **like** *Current*)
     -- Reinitialize by copying the characters of *other*.
     -- (This is also used by *twin*.)
feature(*s*) **from** *STDC_BUFFER*
  -- Comparison
   is_equal (*other*: **like** *Current*): *BOOLEAN*
     -- Is *other* attached to an object considered equal to
     -- current object ?
feature(*s*) **from** *STDC_BUFFER*
  -- Access
   resource_usage_can_be_increased: *BOOLEAN*
     -- Can the number of allocated resources increased with *capacity*?
feature(*s*) **from** *STDC_BUFFER*
  -- Copy data internally or externally

copy_from (*source*: *STDC_BUFFER*; *src_offset, dest_offset, bytes*: *INTEGER*)
   -- Move data from another buffer into ourselves.
   -- Start at offset *src_offset*, into
   -- offset *dest_offset*, moving *bytes* bytes
   -- Memory may overlap.
memory_copy (*source*: *POINTER*; *src_offset*: *INTEGER*; *dest_offset, bytes*: *INTEGER*)
   -- Copy data from *source*, offset *src_offset*, to location
   -- *dest_offset* in this buffer, for *bytes* bytes.
   -- Memory may not overlap, use *move* to copy within buffer
   -- or *memory_move* to copy from potentially overlapping buffer.
memory_move (*source*: *POINTER*; *src_offset*: *INTEGER*; *dest_offset, bytes*: *INTEGER*)
   -- Copy data from *source*, offset *src_offset*, to location
   -- *dest_offset* in this buffer, for *bytes* bytes.
   -- Memory may overlap.
move (*src_offset, dest_offset*: *INTEGER*; *bytes*: *INTEGER*)
   -- Move data around in buffer itself from offset *src_offset* to
   -- offset *dest_offset*, moving *bytes* bytes.
   -- Memory may overlap.
feature(*s*) **from** *STDC_BUFFER*
  -- Access
handle: *POINTER*
   -- Alias for *ptr*
feature(*s*) **from** *STDC_BUFFER*
  -- Set/get bytes (8-bit data)
peek_uint8 (*index*: *INTEGER*): *INTEGER*
   -- consider memory an array of 8 bit values.
infix "@" (*index*: *INTEGER*): *INTEGER*
   -- consider memory an array of 8 bit values.
poke_uint8 (*index, value*: *INTEGER*)
peek_int8 (*index*: *INTEGER*): *INTEGER*
   -- consider memory an array of 8 bit values.
poke_int8 (*index, value*: *INTEGER*)
feature(*s*) **from** *STDC_BUFFER*
  -- Set/get integers (16-bit data)
peek_int16 (*index*: *INTEGER*): *INTEGER*
   -- Read signed 16 bit value at offset *index* in native
   -- endian format.
peek_int16_native (*index*: *INTEGER*): *INTEGER*
   -- Read signed 16 bit value at offset *index* in native
   -- endian format.
peek_uint16 (*index*: *INTEGER*): *INTEGER*
   -- Read unsigned 16 bit value at offset *index* in native format.
peek_uint16_native (*index*: *INTEGER*): *INTEGER*
   -- Read unsigned 16 bit value at offset *index* in native format.
peek_int16_big_endian (*index*: *INTEGER*): *INTEGER*
   -- Read 16 bit value at offset *index* in big endian format.
peek_uint16_big_endian (*index*: *INTEGER*): *INTEGER*
   -- Read 16 bit value at offset *index* in big endian format.
peek_int16_little_endian (*index*: *INTEGER*): *INTEGER*

    -- Read 16 bit value at offset *index* in little endian format.

   peek_uint16_little_endian (*index*: *INTEGER*): *INTEGER*

    -- Read 16 bit value at offset *index* in little endian format.

   poke_int16 (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 16 bit value at offset *index*, in native endian format.

   poke_int16_native (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 16 bit value at offset *index*, in native endian format.

   poke_int16_big_endian (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 16 bit value at offset *index*, in big endian format.

   poke_int16_little_endian (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 16 bit value at offset *index*, in little endian format.

feature(*s*) **from** *STDC_BUFFER*

  -- Set/get integers (32-bit data)

   peek_int32_native (*index*: *INTEGER*): *INTEGER*

    -- Read 32 bit value at offset *index*, assume its byte order

    -- is native, and return it.

   peek_integer (*index*: *INTEGER*): *INTEGER*

    -- Read 32 bit value at offset *index*, assume its byte order

    -- is native, and return it.

   peek_int32_big_endian (*index*: *INTEGER*): *INTEGER*

    -- Read 32 bit value at offset *index*, assume its byte order

    -- is big endian, and return it in native format.

   peek_int32_little_endian (*index*: *INTEGER*): *INTEGER*

    -- Read 32 bit value at offset *index*, assume its byte order

    -- is little endian, and return it in native format.

   peek_uint32_native (*index*: *INTEGER*): *INTEGER*

    -- Read 32 bit unsigned int at offset *index*, assume native

    -- byte order.

   peek_uint32_big_endian (*index*: *INTEGER*): *INTEGER*

    -- Read 32 bit unsigned int at offset *index*, assume its

    -- byte order is big endian, and return it in native format.

   peek_uint32_little_endian (*index*: *INTEGER*): *INTEGER*

    -- Read 32 bit unsigned int at offset *index*, assume its

    -- byte order is big endian, and return it in native format.

   poke_integer (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 32 bit value at offset *index*, in native endian format.

   poke_int32_native (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 32 bit value at offset *index*, in native endian format.

   poke_int32_big_endian (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 32 bit value at offset *index*, in big endian format.

   poke_int32_little_endian (*index*: *INTEGER*; *value*: *INTEGER*)

    -- Write 32 bit value at offset *index*, in little endian format.

feature(*s*) **from** *STDC_BUFFER*

  -- Set/get integers (64-bit data)

   peek_int64_native (*index*: *INTEGER*): *INTEGER_64*

    -- Read 64 bit value at offset *index*, assume its byte order

    -- is native, and return it.

   peek_integer_64 (*index*: *INTEGER*): *INTEGER_64*

    -- Read 64 bit value at offset *index*, assume its byte order

        -- is native, and return it.

peek__int64__big__endian (*index*: *INTEGER*): *INTEGER__64*
    -- Read 64 bit int at offset *index*, assume its
    -- byte order is big endian, and return it in native format.

peek__int64__little__endian (*index*: *INTEGER*): *INTEGER__64*
    -- Read 64 bit int at offset *index*, assume its
    -- byte order is little endian, and return it in native format.

poke__integer__64 (*index*: *INTEGER*; *value*: *INTEGER__64*)
    -- Write 64 bit value at offset *index*, in native endian format.

poke__int64__native (*index*: *INTEGER*; *value*: *INTEGER__64*)
    -- Write 64 bit value at offset *index*, in native endian format.

poke__int64__big__endian (*index*: *INTEGER*; *value*: *INTEGER__64*)
    -- Write 64 bit value at offset *index*, in big endian format.

poke__int64__little__endian (*index*: *INTEGER*; *value*: *INTEGER__64*)
    -- Write 64 bit value at offset *index*, in little endian format.

feature(*s*) **from** *STDC__BUFFER*
  -- Set/get characters

append__to__string (*dest*: *STRING*; *start__index, end__index*: *INTEGER*)
    -- Append all characters from *start__index* to *end__index*
    -- inclusive to *dest*.

peek__character (*index*: *INTEGER*): *CHARACTER*
    -- Return value at *index* as an 8-bit character.

poke__character (*index*: *INTEGER*; *value*: *CHARACTER*)
    -- Set character at *index* index to *value*.

put__character (*c*: *CHARACTER*; *index*: *INTEGER*)
    -- Set character at *index* index to *value*.
    -- Same as *peek__character* with more Eiffel like parameter order.

put__string (*s*: *STRING*; *a__start__index, an__end__index*: *INTEGER*)
    -- Put *s* starting at index *start__index*. *s* is written up
    -- to *end__index* or when there are no more characters in
    -- *s*.

put__to__string (*dest*: *STRING*; *pos, start__index, end__index*: *INTEGER*)
    -- Put characters from *start__index* to *end__index* inclusive
    -- in *dest* starting at position *pos*.
    -- Useful for Gobo character buffers.

c__substring__with__string (*dest*: *STRING*; *start__index, end__index*: *INTEGER*)
    -- As *c__substring* but used *dest* as the destination.

c__substring (*start__index, end__index*: *INTEGER*): *STRING*
    -- Create a substring containing all characters from
    -- *start__index* up to encountering a %U or when *end__index* is
    -- reached, whatever happens first.

substring (*start__index, end__index*: *INTEGER*): *STRING*
    -- Create a substring containing all characters
    -- from start_index to end_index inclusive.

feature(*s*) **from** *STDC__BUFFER*
  -- Fill

fill__at (*start__index, a__count*: *INTEGER*; *byte*: *INTEGER*)
    -- Starting at position *start__index*, write *byte* for *a__count* bytes

feature(*s*) **from** *STDC__BUFFER*

-- Searching
locate_character (*other*: *CHARACTER*; *start_index*: *INTEGER*): *INTEGER*
   -- Return index of *other* in buffer, or -1.
   -- Search begins at *start_index*.
locate_string (*other*: *STRING*; *start_index*: *INTEGER*): *INTEGER*
   -- Does buffer contain other?
   -- Returns index where *other* is found.
   -- Returns -1 if not found
   -- searching starts at position *start_index*
feature(*s*) **from** *STDC_BUFFER*
-- Status
is_valid_index (*index*: *INTEGER*): *BOOLEAN*
is_valid_range (*from_index, to_index*: *INTEGER*): *BOOLEAN*
   -- Is *from_index..to_index* a valid and meaningfull range?
invariant
accessing_real_singleton: *security_is_real_singleton*;
valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
capacity_not_negative: *capacity* $>= 0$;
valid_capacity: *is_allocated* $=$ (*capacity* > 0);
open_implies_handle_assigned: *is_allocated* $=$ (*ptr* $/=$ *unassigned_value*);
owned_implies_open: *is_owner* **implies** *is_allocated*;
owned_implies_handle_assigned: *is_owner* **implies** *ptr* $/=$ *unassigned_value*;
end *of STDC_BUFFER*

A.3 Short form of STDC_CONSTANTS

class *interface STDC_CONSTANTS*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Error codes
   edom: *INTEGER*
     -- Math argument out of domain of function
   erange: *INTEGER*
     -- Math result not representable
   emfile: *INTEGER*
     -- Too many open files
feature(*s*) **from** *STDC_CONSTANTS*
   -- Standard streams
   stream_stdin: *POINTER*
   stream_stdout: *POINTER*
   stream_stderr: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Special characters
   const_eof: *INTEGER*
     -- signals EOF
feature(*s*) **from** *STDC_CONSTANTS*
   -- I/O buffering
   iofbf: *INTEGER*
     -- full buffering
   iolbf: *INTEGER*
     -- line buffering
   ionbf: *INTEGER*
     -- no buffering
feature(*s*) **from** *STDC_CONSTANTS*
   -- file positioning
   seek_set: *INTEGER*
   seek_cur: *INTEGER*
   seek_end: *INTEGER*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Signal related constants
   sig_dfl: *POINTER*
   sig_err: *POINTER*
   sig_ign: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Signals
   sigabrt: *INTEGER*
   sigfpe: *INTEGER*
     -- erroneous arithmetic operation, such as zero divide or an
     -- operation resulting in overflow
   sigill: *INTEGER*
     -- illegal instruction
   sigint: *INTEGER*
     -- receipt of an interactive attention signal
   sigsegv: *INTEGER*

-- invalid access to storage

sigterm: *INTEGER*

-- Request process to terminate; can be caught or ignored

feature(*s*) **from** *STDC_CONSTANTS*

-- random numbers

rand_max: *INTEGER*

-- maximum value returned by the *random* function

feature(*s*) **from** *STDC_CONSTANTS*

-- category constants

lc_ctype: *INTEGER*

lc_numeric: *INTEGER*

lc_time: *INTEGER*

lc_collate: *INTEGER*

lc_monetary: *INTEGER*

lc_all: *INTEGER*

feature(*s*) **from** *STDC_CONSTANTS*

-- various

clocks_per_sec: *INTEGER*

feature(*s*) **from** *STDC_CONSTANTS*

-- exit codes

exit_failure: *INTEGER*

-- exit status when something has gone wrong

exit_success: *INTEGER*

-- exit status upon success

end *of STDC_CONSTANTS*

A.4 Short form of STDC_CURRENT_PROCESS

class *interface STDC_CURRENT_PROCESS*
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
  -- The singleton, available to any because its used in preconditions
  security: *STDC_SECURITY*
    -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
  -- Access
  errno: *STDC_ERRNO*
    -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
  -- Status
  raise_exception_on_error: *BOOLEAN*
    -- Should an exception be raised when an error occurs?
    -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
  -- Change
  set_default_action_on_error
    -- Use *security.error_handling.exceptions_enabled* to
    -- determine if an exception should be raised when a C call
    -- returns an error.
  set_raise_exception_on_error
    -- Always raise an exception when a C call returns an error.
  set_continue_on_error
    -- Never raise an exception when a C call returns an error.
  inherit_error_handling (*an_instance*: *STDC_BASE*)
    -- Handle errors like *an_instance*
feature(*s*) **from** *ARGUMENTS*
  command_name: *STRING*
feature(*s*) **from** *CAPI_TIME*
  -- Standard C binding
  current_time: *INTEGER*
    -- The current calendar time in seconds since the epoch
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Process standard input/output/error
  stdin: *STDC_TEXT_FILE*
  stdout: *STDC_TEXT_FILE*
  stderr: *STDC_TEXT_FILE*
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Various
  clock: *INTEGER*
    -- Approximation of processor time used by the program, or -1
    -- if unknown
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Random numbers
  random: *INTEGER*
    -- Returns a pseudo-random integer between 0 and *RAND_MAX*.
  set_random_seed (*a_seed*: *INTEGER*)

-- Sets *a_seed* as the seed for a new sequence of
-- pseudo-random integers to be returned by *random*. These
-- sequences are repeatable by calling *set_random_seed* with
-- the same seed value. If no seed value is provided, the
-- *random* function is automatically seeded with a value of
-- 1.

feature(*s*) **from** *STDC_CURRENT_PROCESS*
-- Global locale
locale: *STRING*
-- Current locale
numeric_format: *STDC_LOCALE_NUMERIC*
-- Various information for formatting numbers and monetary
-- quantities
set_locale (*category*: *INTEGER*; *new_locale*: *STRING*)
-- Set given locale to *new_locale*. *new_locale* is either a
-- well-known constant like "C" or "da_DK" or an opaque
-- string that was returned by another call of *setlocale*.
set_c_locale
-- Set locale to the Standard C locale (the default).
set_native_decimal_point
-- Set the decimal point character using the LC_NUMERIC
-- environment variable.
set_native_locale
-- Set entire locale to the natives setting which is
-- determend by environment variables like LC_NUMERIC,
-- LC_COLLATE, LC_CTYPE etc.
set_native_time
-- Set time display to the natives setting using the LC_TIME
-- environment variable.

invariant
accessing_real_singleton: *security_is_real_singleton*;
valid_error_action: *error_action >= 0* **and** *error_action <= 2*;
end *of STDC_CURRENT_PROCESS*

A.5   Short form of STDC_ENV_VAR

class *interface STDC_ENV_VAR*
creation
   make (*a_name*: *STRING*)
feature(*s*) **from** *STDC_ENV_VAR*
   -- Initialization
   make (*a_name*: *STRING*)
feature(*s*) **from** *STDC_ENV_VAR*
   -- Access
   exist: *BOOLEAN*
     -- Is this environment variable defined?
   name: *STRING*
     -- Name of environment variable.
   value: *STRING*
     -- Current value of environment variable.
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
end *of STDC_ENV_VAR*

A.6 Short form of STDC_FILE

STDC_FILE is a deferred class. Use STDC_TEXT_FILE for accessing and creating text files, or STDC_BINARY_FILE for binary files.

deferred class *interface STDC_FILE*
feature(*s*) **from** *STDC_FILE*
   -- Initialization
   create_read_write (*path*: *STRING*)
      -- Open file for update (reading and writing). If the file
      -- already exists, it is truncated to zero length.
      -- So permissions seem to remain.
   create_write (*path*: *STRING*)
      -- Create new file for writing. If the file already exists,
      -- it is truncated to zero length.
      -- So permissions seem to remain.
   open (*path, a_mode*: *STRING*)
      -- Open file in given *a_mode*.
   open_append (*path*: *STRING*)
      -- Append to exiting file or create file if it does not exist.
   open_read (*path*: *STRING*)
      -- Open file for reading.
   open_read_write (*path*: *STRING*)
      -- Open file for reading and writing.
   open_write (*path*: *STRING*)
      -- Open file for writing.
feature(*s*) **from** *STDC_FILE*
   -- Work with existing streams
   attach_to_stream (*a_stream*: *POINTER*; *a_mode*: *STRING*)
      -- Attach to *a_stream*. Does not become owner of stream so
      -- it will not close on *close* or when garbage collected.
feature(*s*) **from** *STDC_FILE*
   -- Reopen
   reopen (*a_path, a_mode*: *STRING*)
      -- Closes and then opens a stream.
feature(*s*) **from** *STDC_FILE*
   -- Control over buffering
   flush
      -- Updates this stream
   setbuf (*buffer*: *POINTER*)
      -- Determines how the stream will be buffered
      -- gives you a fully buffered input and output.
      -- Not sure: buffer should have at least BUFSIZ bytes?
      -- No operation should yet been performed on this file
      -- *buffer* = default_pointer: default buffer will be allocated
      -- *buffer* /= default_pointer implies buffer size = BUFSIZ
   set_buffer (*buffer*: *POINTER*)
      -- Determines how the stream will be buffered
      -- gives you a fully buffered input and output.
      -- Not sure: buffer should have at least BUFSIZ bytes?

          -- No operation should yet been performed on this file
          -- *buffer* = default_pointer: default buffer will be allocated
          -- *buffer* /= default_pointer implies buffer size = BUFSIZ
      set_full_buffering (*buffer*: *POINTER*; *size*: *INTEGER*)
          -- Determines buffering for a stream.
          -- If *buffer* is *default_pointer*, a buffer of *size* bytes
          -- will be allocated by this routine.
      set_line_buffering (*buffer*: *POINTER*; *size*: *INTEGER*)
          -- Determines buffering for a stream.
          -- If *buffer* is *default_pointer, a buffer of size* bytes
          -- will be allocated by this routine.
      set_no_buffering
          -- Turn buffering off.
  feature(*s*) **from** *STDC__FILE*
      -- read, C like
      last_byte: *INTEGER*
          -- Last read character of *get_character.*
          -- Can be negative, so is more a last_shortint or so!
      getc
          -- Reads a C unsigned char and converts it to an integer,
          -- the result is left in *last_byte.*
          -- This function probably can be used to read a single
          -- byte.
      get_character
          -- Reads a C unsigned char and converts it to an integer,
          -- the result is left in *last_byte.*
          -- This function probably can be used to read a single
          -- byte.
      read (*buf*: *POINTER*; *offset, bytes*: *INTEGER*)
          -- Read chunk, set *last_read. offset* determines how far
          -- in *buf* you want to start writing.
  feature(*s*) **from** *STDC__FILE*
      -- Write, C like
      putc (*c*: *INTEGER*)
          -- Write a single character.
      write (*buf*: *POINTER*; *offset, bytes*: *INTEGER*)
          -- write *bytes* bytes from *buf* at offset *offset*
          -- we do not really care if offset is positive or negative...
  feature(*s*) **from** *STDC__FILE*
      -- Access
      last_boolean: *BOOLEAN*
          -- last boolean read by *read_boolean*
      last_double: *DOUBLE*
          -- last double lread by *read_double*
      last_integer: *INTEGER*
      last_real: *REAL*
          -- last real read by *read_real*
      max_line_length: *INTEGER*
          -- Maximum line length used in *read_line*

      mode: *STRING*
        -- Mode in which the file is opened/created.
   feature(*s*) **from** *STDC_FILE*
    -- Input
    read_boolean
      -- Attempt to read back a boolean written by *write_boolean*.
    read_buffer (*buf*: *STDC_BUFFER*; *offset, bytes*: *INTEGER*)
      -- More safe version of *read* in case you have a
      -- STDC_BUFFER object. Read starts at *offset* bytes in *buf*.
      -- Check *last_read* for number of bytes actually read.
    read_double
    read_character
      -- Read a single character and set *last_character*.
      -- If end-of-file encountered, *eof* is True.
    read_integer
    read_line
      -- Read characters from input stream until a line separator
      -- or end of file is reached. Make the characters that have
      -- been read available in *last_string* and discard the line
      -- separator characters from the input stream.
      -- Reads a maximum of *max_line_length* characters per line.
      -- The line should not have a %U character in it, because
      -- that is treated as end-of-file.
    read_new_line
      -- Read a line separator from input file.
      -- Make the characters making up the recognized
      -- line separator available in *last_string*,
      -- or make *last_string* empty and leave the
      -- input file unchanged if no line separator
      -- was found.
    read_real
    read_string (*nb*: *INTEGER*)
      -- Read at most *nb* characters from input stream.
      -- Make the characters that have actually been read
      -- available in *last_string*.
      -- The input stream should not contain %U characters.
   feature(*s*) **from** *STDC_FILE*
    -- write, Eiffel like
    put (*any*: *ANY*)
      -- Write object as string.
    put_buffer (*buf*: *STDC_BUFFER*; *offset, bytes*: *INTEGER*)
      -- more safe version of *write* in case you have a
      -- STDC_BUFFER object
      -- Check *last_written* for number of bytes actually written,
      -- if you use asynchronous writing.
    write_buffer (*buf*: *STDC_BUFFER*; *offset, bytes*: *INTEGER*)
      -- more safe version of *write* in case you have a
      -- STDC_BUFFER object
      -- Check *last_written* for number of bytes actually written,

      -- if you use asynchronous writing.
    put_boolean (*b*: *BOOLEAN*)
      -- Write "True" to output stream if
      -- *b* is true, "False" otherwise.
    write_boolean (*b*: *BOOLEAN*)
    put_character (*c*: *CHARACTER*)
      -- Write a single character.
    write_character (*c*: *CHARACTER*)
      -- Write a single character.
    put_double (*d*: *DOUBLE*)
      -- Write a double in Standard C %f format.
    write_double (*d*: *DOUBLE*)
      -- Write a double in Standard C %f format.
    put_integer (*i*: *INTEGER*)
      -- Write an integer in Standard C %d format.
    write_integer (*i*: *INTEGER*)
      -- Write an integer in Standard C %d format.
    put_real (*r*: *REAL*)
      -- Write a real in Standard C %f format.
    write_real (*r*: *REAL*)
      -- Write a real in Standard C %f format.
    put_string (*a_string*: *STRING*)
      -- Write *a_string* to stream.
      -- Because the way this feature is written (it supports
      -- writing the NULL byte), it is probably a very good idea to
      -- turn on buffering, see *set_full_buffering* or
      -- *set_line_buffering*.
    write_string (*s*: *STRING*)
    puts (*s*: *STRING*)
feature(*s*) **from** *STDC_FILE*
  -- Unreading
  ungetc (*c*: *INTEGER*)
    -- Pushes *c* back to the stream. Only one push back is guaranteed.
    -- Note that file positioning functions discard any
    -- pushed-back characters.
  unread_character (*an_item*: *CHARACTER*)
    -- Put *an_item* back in input stream. Only one push back is
    -- guaranteed.
    -- This item will be read first by the next
    -- call to a read routine.
    -- Note that file positioning functions discard any
    -- pushed-back characters.
feature(*s*) **from** *STDC_FILE*
  -- File position
  get_position: *STDC_FILE_POSITION*
    -- Get the current position. Use *set_position* to return to
    -- this saved position
  rewind
    -- Sets the file position to the beginning of the file.

    seek (*offset*: *INTEGER*)
      -- Set file position to given absolute *offset*.
    seek_from_current (*offset*: *INTEGER*)
      -- Set file position relative to current position.
    seek_from_end (*offset*: *INTEGER*)
      -- Set file position relative to end of file.
    set_position (*a_position*: *STDC_FILE_POSITION*)
      -- Set the current file position.
    tell: *INTEGER*
      -- The current position
feature(*s*) **from** *STDC_FILE*
  -- Other
    clearerr
      -- Clears end-of-file and error indicators for a stream.
    clear_error
      -- Clears end-of-file and error indicators for a stream.
feature(*s*) **from** *STDC_FILE*
  -- Status
    end_of_input: *BOOLEAN*
      -- Is end-of-file encountered by getc or is the end-of-file indicator
      -- is set?
    error: *BOOLEAN*
      -- Is the error indicator is set?
    is_binary_mode_specification (*a_mode*: *STRING*): *BOOLEAN*
      -- Is the last character of a_mode equal to b?
    is_text_mode_specification (*a_mode*: *STRING*): *BOOLEAN*
      -- Is the last character of a_mode equal to t?
    is_valid_mode (*a_mode*: *STRING*): *BOOLEAN*
      -- Is *a_mode* a valid mode specification for *Current*?
      ensure
        not_empty: *Result* **implies** *a_mode* /= *Void* **and then not** *a_mode.is_empty*
    resource_usage_can_be_increased: *BOOLEAN*
      -- Is it allowed to open another file?
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
  open_in_sync: *is_open_read* **or** *is_open_write* **implies** *is_open*; -- The reverse is not
true, for examples sockets can be
 -- closed for reading/writing, but still open.
  capacity_not_negative: *capacity* >= *0*;
  valid_capacity: *is_open* = (*capacity* > *0*);
  open_implies_handle_assigned: *is_open* = (*stream* /= *unassigned_value*);
  owned_implies_open: *is_owner* **implies** *is_open*;
  owned_implies_handle_assigned: *is_owner* **implies** *stream* /= *unassigned_value*;
  last_string_valid: *last_string* /= *Void*;
  gets_buf_valid: *gets_buf* /= *Void*;
end *of* **deferred** *STDC_FILE*

A.7 Short form of STDC_FILE_SYSTEM

class *interface STDC_FILE_SYSTEM*
feature(*s*) **from** *STDC_FILE_SYSTEM*
   -- Path names
   expand_path (*a_path*: *STRING*): *STDC_PATH*
     -- returns a new path
feature(*s*) **from** *STDC_FILE_SYSTEM*
   -- Rename files/directories, remove files/directories
   remove_file (*a_path*: *STRING*)
     -- Removes a file from a directory.
     -- For Standard C, its implementation defined what
     -- remove_file does if file is opened by some process
     -- (*remove_file* fails on Windows for example).
     -- doesnt remove a directory.
   rename_to (*current_path, new_path*: *STRING*)
     -- Rename a file or a directory.
     -- *new_path* should not be an existing path.
feature(*s*) **from** *STDC_FILE_SYSTEM*
   -- Accessibility of files
   is_modifiable (*a_path*: *STRING*): *BOOLEAN*
     -- Is *a_path* readable and writable by this program?
     -- Does this by attempting to open *a_path* file read/write.
   is_readable (*a_path*: *STRING*): *BOOLEAN*
     -- Is *a_path* readable by this program?
     -- Does this by attempting to open *a_path* file read-only.
feature(*s*) **from** *STDC_FILE_SYSTEM*
   -- File and string
   write_string_to_file (*s, a_file_name*: *STRING*)
     -- Write *s* to file *a_file_name*.
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
end *of STDC_FILE_SYSTEM*

A.8 Short form of STDC_SECURITY

class *interface STDC_SECURITY*
feature(*s*) **from** *STDC_SECURITY*
   -- Modes
  make_allow_all
    -- Just allow everything.
  make_allow_sandbox
    -- Allow very little, use for setuid root programs.
feature(*s*) **from** *STDC_SECURITY*
   -- The security aspects
  cpu: *STDC_SECURITY_CPU*
  error_handling: *STDC_SECURITY_ERROR_HANDLING*
  files: *STDC_SECURITY_FILES*
  memory: *STDC_SECURITY_MEMORY*
feature(*s*) **from** *STDC_SECURITY*
   -- Various
  assert_once_memory_allocated
    -- Make sure that certain once functions in STDC_BASE are
    -- called. These once functions are called when an error
    -- occurs, at that time there might not be memory left to
    -- create them.
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action $>= 0$* **and** *error_action $<= 2$*;
  remain_single: *Current = singleton*;
end *of STDC_SECURITY*

A.9 Short form of STDC_SIGNAL

class *interface STDC_SIGNAL*
creation
  make (*a_value*: *INTEGER*)
feature(*s*) **from** *STDC_SIGNAL*
  -- creation
  make (*a_value*: *INTEGER*)
feature(*s*) **from** *STDC_SIGNAL*
  -- set signal properties, make effective with *apply*
  apply
    -- Make changes effective.
  set_default_action
    -- Install signal-specific default action.
    -- Call *apply* to make changes effective.
  set_ignore_action
    -- Set action to ignore signal.
    -- Call *apply* to make changes effective.
  set_handler (*a_handler*: *STDC_SIGNAL_HANDLER*)
    -- Install ones own signal handler.
feature(*s*) **from** *STDC_SIGNAL*
  -- signal functions
  raise
    -- Raise the signal.
feature(*s*) **from** *STDC_SIGNAL*
  -- signal state
  is_ignorable: *BOOLEAN*
    -- All signals Standard C knows about are ignorable...
  value: *INTEGER*
    -- the signal
invariant
  --accessing_real_singleton: signal_switch_is_real_singleton
  -- Gives crash with ISE Eiffel
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
  valid_signal_value: *value* >= *1*;
end *of STDC_SIGNAL*

A.10 Short form of STDC_SIGNAL_HANDLER

deferred class *interface STDC_SIGNAL_HANDLER*
end *of* **deferred** *STDC_SIGNAL_HANDLER*

A.11 Short form of STDC_SYSTEM

class *interface STDC_SYSTEM*
feature(*s*) **from** *STDC_SYSTEM*
   -- run-time determined queries
   is_shell_available: *BOOLEAN*
      -- Return True if command interpreter is available
   is_windows: *BOOLEAN*
      -- Are we running on the Windows platform?
      -- Note that this is false when using cygwin as for all
      -- intends and purposes cygwin is unix to a program that
      -- compiled with it.
feature(*s*) **from** *STDC_SYSTEM*
   -- Compile time determined queries
   clocks_per_second: *INTEGER*
      -- Number per second of the value returned by the *clock* function
feature(*s*) **from** *STDC_SYSTEM*
   -- Time zone
   time_zone_seconds: *INTEGER*
      -- Number of seconds to add to UTC to arrive at the time for
      -- the current time zone
feature(*s*) **from** *STDC_SYSTEM*
   -- Endianess
   is_big_endian: *BOOLEAN*
      -- True if this is a big endian architecture
   is_little_endian: *BOOLEAN*
      -- True if this is a little endian architecture
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
end *of STDC_SYSTEM*

A.12 Short form of STDC_TIME

class *interface STDC_TIME*
creation
   make_date (*a_year, a_month, a_day*: *INTEGER*)
     -- Create a time according to this day, time 00:00:00.
     -- Date is assumed to be a local date.
   make_date_time (*a_year, a_month, a_day, an_hour, a_minute, a_second*: *INTEGER*)
     -- Date is assumed to be a local date.
     -- We assume daylight saving time setting in effect is
     -- available from system.
   make_from_dt_date_time (*a_date_time*: *DT_DATE_TIME_VALUE*)
     -- Make from Gobo date time.
     -- Date is assumed to be a local date.
     -- We assume daylight saving time setting in effect is
     -- available from system.
   make_from_now
     -- Make *value* equal to current unix time.
     -- Afterwards call *to_local* or *to_utc* to turn individual
     -- fields in local time or in utc time.
   make_from_unix_time (*a_value*: *INTEGER*)
     -- *a_value* is a time_t value.
     -- Afterwards call *to_local* or *to_utc* to turn individual
     -- fields in local time or in utc time.
   make_time (*an_hour, a_minute, a_second*: *INTEGER*)
     -- Time is assumed to be a local time.
     -- We assume daylight saving time setting in effect is
     -- available from system.
     -- Day will be January 1, *minimum_year*.
   make_utc_date (*a_year, a_month, a_day*: *INTEGER*)
     -- Create a time according to this day, time 00:00:00.
     -- Date is assumed to be in UTC.
   make_utc_date_time (*a_year, a_month, a_day, an_hour, a_minute, a_second*: *INTEGER*)
     -- Date is assumed to be in UTC.
     -- Conversion to the unix time is done without taking into
     -- account leap seconds, as according to the specification.
   make_utc_time (*an_hour, a_minute, a_second*: *INTEGER*)
     -- Time is assumed to be UTC time at January 1, *minimum_year*.
     -- We assume daylight saving time setting in effect is
     -- available from system.
  feature(*s*) **from** *STDC_TIME*
   -- Initialization
   make_date (*a_year, a_month, a_day*: *INTEGER*)
     -- Create a time according to this day, time 00:00:00.
     -- Date is assumed to be a local date.
   make_date_time (*a_year, a_month, a_day, an_hour, a_minute, a_second*: *INTEGER*)
     -- Date is assumed to be a local date.
     -- We assume daylight saving time setting in effect is
     -- available from system.

make_date_time_without_dst (*a_year, a_month, a_day, an_hour, a_minute, a_second*: *INTEGER*)
    -- Date is assumed to be a date/time without daylight saving
    -- taken into account, such as a UTC based date/time.

make_from_dt_date_time (*a_date_time*: *DT_DATE_TIME_VALUE*)
    -- Make from Gobo date time.
    -- Date is assumed to be a local date.
    -- We assume daylight saving time setting in effect is
    -- available from system.

make_from_now
    -- Make *value* equal to current unix time.
    -- Afterwards call *to_local* or *to_utc* to turn individual
    -- fields in local time or in utc time.

make_from_unix_time (*a_value*: *INTEGER*)
    -- *a_value* is a time_t value.
    -- Afterwards call *to_local* or *to_utc* to turn individual
    -- fields in local time or in utc time.

make_utc_date (*a_year, a_month, a_day*: *INTEGER*)
    -- Create a time according to this day, time 00:00:00.
    -- Date is assumed to be in UTC.

make_utc_date_time (*a_year, a_month, a_day, an_hour, a_minute, a_second*: *INTEGER*)
    -- Date is assumed to be in UTC.
    -- Conversion to the unix time is done without taking into
    -- account leap seconds, as according to the specification.

make_utc_time (*an_hour, a_minute, a_second*: *INTEGER*)
    -- Time is assumed to be UTC time at January 1, *minimum_year*.
    -- We assume daylight saving time setting in effect is
    -- available from system.

feature(*s*) **from** *STDC_TIME*
  -- Make individual time fields valid

is_local_time: *BOOLEAN*
    -- Is time in *local time*?

is_utc_time: *BOOLEAN*
    -- Is the time zone UTC?

is_time_zone_known: *BOOLEAN*
    -- After a make routine, call either *to_local* or *to_utc*.

to_local
    -- Switch time fields to local time based on time in *value*.

to_utc
    -- Switch time fields to utc time based on time in *value*.

feature(*s*) **from** *STDC_TIME*
  -- Manually set individual time fields

set_date (*a_year, a_month, a_day*: *INTEGER*)
    -- Set date part, time remains unchanged, unless daylight
    -- savings has to be taken into account.

set_date_time (*a_year, a_month, a_day, an_hour, a_minute, a_second*: *INTEGER*)
    -- Set individual time fields. Set *value* based on given
    -- fields, assuming that it is a local time.
    -- We assume daylight saving time setting in effect (or not)

      -- has been set.

    set_dst_to_current

      -- Let system figure out if daylight saving time is in effect.

    set_dst_to_none

      -- Daylight saving time is not in effect.

    set_dst_in_effect

      -- Daylight saving time is in effect.

    set_time (*an_hour, a_minute, a_second*: *INTEGER*)

      -- Set time part, date remains unchanged unless daylight

      -- savings has to be taken into account.

    to_dos_seconds

      -- Make sure the seconds are divisible by two, a value DOS

      -- and clones like Windows NT like.

  feature(*s*) **from** *STDC_TIME*

    -- Individual time fields, need call to *to_local* or *to_utc*

    year: *INTEGER*

    month: *INTEGER*

    day: *INTEGER*

      -- Day of the month.

    weekday: *INTEGER*

      -- Days since Sunday.

    day_of_year: *INTEGER*

      -- Days since January 1st

    hour: *INTEGER*

    minute: *INTEGER*

    second: *INTEGER*

    is_daylight_savings_in_effect: *BOOLEAN*

      -- Does the broken down time take into account daylight savings?

    is_daylight_savings_unknown: *BOOLEAN*

      -- Do we not know if the broken time includes daylight saving?

  feature(*s*) **from** *STDC_TIME*

    -- Time as string

    short_weekday_name: *STRING*

      -- Abbreviated weekday name

    weekday_name: *STRING*

      -- Full weekday name

    short_month_name: *STRING*

      -- Abbreviated month name

    month_name: *STRING*

      -- Full month name

    format (*format_str*: *STRING*): *STRING*

      -- Formatted date/time according to *format_str*. See

      -- man strftime for details.

    default_format: *STRING*

      -- Time as string of the form "Mon Apr 17 21:49:20 2000"

    local_date_string: *STRING*

      -- Date part in format local to current country.

    local_time_string: *STRING*

      -- Time part in format local to current country.

rfc_date_string: *STRING*
    -- RFC 1123 (same as RFC 822) style date;
    -- i.e. Tue, 15 Nov 1994 08:12:31 GMT

feature(*s*) **from** *STDC_TIME*
  -- Date calculations
  is_equal (*other*: **like** *Current*): *BOOLEAN*
    -- Is *other* attached to an object considered equal to
    -- current object ?
  infix "+" (*other*: **like** *Current*): **like** *Current*
    -- Sum with *other*
  infix "-" (*other*: **like** *Current*): **like** *Current*
    -- Creates a new time which is the difference between
    -- *Current* and *Other*
  infix "<" (*other*: **like** *Current*): *BOOLEAN*
    -- Is current object less than *other*?

feature(*s*) **from** *STDC_TIME*
  -- Status
  is_two_digit_year (*a_year*: *INTEGER*): *BOOLEAN*
    -- Is *a_year* a two digit year that can be handled by
    -- *four_digit_year.*
  is_valid_date (*a_year, a_month, a_day*: *INTEGER*): *BOOLEAN*
    -- Do *a_year, a_month* and *a_day* form a date recognized
    -- by this class?
    -- Because this class represents unix dates, only dates
    -- between 1970-Jan-01 UTC and 2038-Jan-19 UTC are valid.
  is_valid_date_and_time (*a_year, a_month, a_day, an_hour, a_minute, a_second*:
INTEGER): *BOOLEAN*
    -- Do *a_year, a_month* and *a_day* form a date that can be
    -- represented by this class?
    -- Because this class represents unix dates, only dates
    -- between 1970-Jan-01 00:00 UTC and 2038-Jan-19 03:14:08 UTC
    -- are valid.
  is_valid_day (*a_year, a_month, a_day*: *INTEGER*): *BOOLEAN*
    -- Is *a_day* a valid day given year and month.
  is_valid_time (*an_hour, a_minute, a_second*: *INTEGER*): *BOOLEAN*
    -- Do *an_hour, a_minute* and *a_second* form a valid 24
    -- hour clock time?

feature(*s*) **from** *STDC_TIME*
  -- Access
  current_year: *INTEGER*
    -- Current year.
  four_digit_year (*a_year*: *INTEGER*): *INTEGER*
    -- Return a four digit year given a possibly two digit year.
  hash_code: *INTEGER*
    -- The hash-code value of *Current.*
  minimum_year: *INTEGER*
    -- The minimum year for the current platform.
    -- For POSIX is 1970, for Windows is 1980.
  maximum_year: *INTEGER*

       -- The maximum Epoch year.
    value: *INTEGER*
       -- Time in seconds since January 1, 1970.
  feature(*s*) **from** *STDC_TIME*
    -- Conversion
    as_dt_date_time: *DT_DATE_TIME*
       -- Date time in Gobo date time format.
       -- Always returns a new object.
  invariant
    accessing_real_singleton: *security_is_real_singleton*;
    valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
    tm_not_void: *tm* $/=$ *Void*;
    tm_has_proper_capacity: *tm.capacity* $>=$ *posix_tm_size*;
    value_not_negative: *value* $>= 0$;
    my_time_zone_valid: *my_time_zone* $= 0$ **or else** *my_time_zone* $=$ *utc_time_zone*
  or else *my_time_zone* $=$ *local_time_zone*;
  end *of STDC_TIME*

B
Short list-
ing of ab-
stract classes

An abstract class is somewhat above the Standard C classes, and between the features you get when you use a POSIX or Windows class. It is mainly aimed at users who want to write software usable on Unix and Windows, and who do not want to use a POSIX emulator.

You never use an abstract class directly, always use the corresponding effective EPX_XXXX, for which there is a variant in the `src/posix` or `src/windows` directory.

B.1   Short form of ABSTRACT_CURRENT_PROCESS

```
deferred class interface ABSTRACT_CURRENT_PROCESS
feature(s) from ARGUMENTS
   command_name: STRING
feature(s) from CAPI_TIME
   -- Standard C binding
   current_time: INTEGER
      -- The current calendar time in seconds since the epoch
feature(s) from STDC_SECURITY_ACCESSOR
   -- The singleton, available to any because its used in preconditions
   security: STDC_SECURITY
      -- Singleton entry point for security.
feature(s) from STDC_BASE
   -- Access
   errno: STDC_ERRNO
      -- Access to the variable that contains the error that occurred.
feature(s) from STDC_BASE
   -- Status
   raise_exception_on_error: BOOLEAN
      -- Should an exception be raised when an error occurs?
      -- If not, you have to check errno for any errors.
feature(s) from STDC_BASE
   -- Change
   set_default_action_on_error
      -- Use security.error_handling.exceptions_enabled to
```

-- determine if an exception should be raised when a C call

-- returns an error.

set_raise_exception_on_error

-- Always raise an exception when a C call returns an error.

set_continue_on_error

-- Never raise an exception when a C call returns an error.

inherit_error_handling (*an_instance*: *STDC_BASE*)

-- Handle errors like *an_instance*

feature(*s*) **from** *STDC_CURRENT_PROCESS*

-- Process standard input/output/error

stdin: *STDC_TEXT_FILE*

stdout: *STDC_TEXT_FILE*

stderr: *STDC_TEXT_FILE*

feature(*s*) **from** *STDC_CURRENT_PROCESS*

-- Various

clock: *INTEGER*

-- Approximation of processor time used by the program, or -1

-- if unknown

feature(*s*) **from** *STDC_CURRENT_PROCESS*

-- Random numbers

random: *INTEGER*

-- Returns a pseudo-random integer between 0 and *RAND_MAX*.

set_random_seed (*a_seed*: *INTEGER*)

-- Sets *a_seed* as the seed for a new sequence of

-- pseudo-random integers to be returned by *random*. These

-- sequences are repeatable by calling *set_random_seed* with

-- the same seed value. If no seed value is provided, the

-- *random* function is automatically seeded with a value of

-- 1.

feature(*s*) **from** *STDC_CURRENT_PROCESS*

-- Global locale

locale: *STRING*

-- Current locale

numeric_format: *STDC_LOCALE_NUMERIC*

-- Various information for formatting numbers and monetary

-- quantities

set_locale (*category*: *INTEGER*; *new_locale*: *STRING*)

-- Set given locale to *new_locale*. *new_locale* is either a

-- well-known constant like "C" or "da_DK" or an opaque

-- string that was returned by another call of *setlocale*.

set_c_locale

-- Set locale to the Standard C locale (the default).

set_native_decimal_point

-- Set the decimal point character using the LC_NUMERIC

-- environment variable.

set_native_locale

-- Set entire locale to the natives setting which is

-- determend by environment variables like LC_NUMERIC,

-- LC_COLLATE, LC_CTYPE etc.

set_native_time
  -- Set time display to the natives setting using the LC_TIME
  -- environment variable.
feature(*s*) **from** *ABSTRACT_PROCESS*
-- Process properties
pid: *INTEGER*
  -- Process identifier, unique for this process
is_pid_valid: *BOOLEAN*
  -- Is *pid* valid?
feature(*s*) **from** *ABSTRACT_PROCESS*
-- Signal this process
terminate
  -- Attempt to gracefully terminate this process.
  require
    valid_pid: *is_pid_valid*
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- Access
effective_user_name: *STRING*
  -- Name of the user currently associated with the current
  -- thread
  ensure
    name_not_void: *Result* /= *Void*
full_command_name: *STRING*
  -- *command_name* with fully qualified path;
  -- An empty string is returned in case *command_name* is
  -- empty. As any program can setup the arguments passed to
  -- another program, an empty *command_name* is a possibility.
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- Every process also has standard file descriptors which might not be compatible with
stdin/stdout/stderr (Windows)
fd_stdin: *ABSTRACT_FILE_DESCRIPTOR*
  ensure
    fd_stdin_not_void: *Result* /= *Void*;
    not_owner: **not** *Result.is_owner*
fd_stdout: *ABSTRACT_FILE_DESCRIPTOR*
  ensure
    fd_stdout_not_void: *Result* /= *Void*;
    not_owner: **not** *Result.is_owner*
fd_stderr: *ABSTRACT_FILE_DESCRIPTOR*
  ensure
    fd_stderr_not_void: *Result* /= *Void*;
    not_owner: **not** *Result.is_owner*
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- Sleeping
millisleep (*a_milliseconds*: *INTEGER*)
  -- Sleep for *a_milliseconds* milliseconds. Due to timer
  -- resolution issues, the minimum resolution might be in the
  -- order of 10ms or higher.
  require

        milliseconds_not_negative: *a_milliseconds* >= *0*
    sleep (*seconds*: *INTEGER*)
      -- Delays process execution up to *seconds*. Can return early
      -- if interrupted. Check *unslect_seconds*
    unslept_seconds: *INTEGER*
      -- The number of seconds still to sleep, before being
      -- interrupted; it is set by *sleep*. If it is zero, no
      -- interrupt occurred and process slept for the allotted
      -- time.
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
end *of* **deferred** *ABSTRACT_CURRENT_PROCESS*

B.2 *Short form of ABSTRACT__EXEC__PROCESS*

deferred class *interface ABSTRACT__EXEC__PROCESS*
feature(*s*) **from** *ABSTRACT__EXEC__PROCESS*
  -- (re)set arguments
  has__void__argument (*a__arguments*: *ARRAY*[*STRING*]): *BOOLEAN*
    -- Is one of the items in *a__arguments* Void?
  set__arguments (*a__arguments*: *ARRAY*[*STRING*])
feature(*s*) **from** *ABSTRACT__EXEC__PROCESS*
  -- i/o capturing
  capture__input: *BOOLEAN*
    -- is input captured on execute?
  capture__output: *BOOLEAN*
    -- is output captured on execute?
  capture__error: *BOOLEAN*
    -- is error captured on execute?
  set__capture__input (*on*: *BOOLEAN*)
  set__capture__output (*on*: *BOOLEAN*)
  set__capture__error (*on*: *BOOLEAN*)
  fd__stdin: *ABSTRACT__FILE__DESCRIPTOR*
    -- Input read by process
  fd__stdout: *ABSTRACT__FILE__DESCRIPTOR*
    -- Output emitted by process
  fd__stderr: *ABSTRACT__FILE__DESCRIPTOR*
    -- Error output from process
feature(*s*) **from** *ABSTRACT__EXEC__PROCESS*
  -- Execute
  execute
    -- Execute *program__name* with arguments *arguments*. After
    -- execution, at some point in time, you have to *wait* or
    -- *wait__for* for this process to terminate.
    require
      not__already__started: *is__terminated*
feature(*s*) **from** *ABSTRACT__EXEC__PROCESS*
  -- Actions that parent may execute
  wait__for (*suspend*: *BOOLEAN*)
    -- Wait for this process to terminate. If *suspend* then we
    -- wait until the information about this process is available,
    -- else we return immediately.
    -- If suspend is False, check *is__terminated* to see
    -- if this child is really terminated.
    require
      pid__refers__to__child: *is__pid__valid*;
      not__terminated: **not** *is__terminated*
    ensure
      stdin__closed: *is__terminated* **implies** *fd__stdin* = *Void* **or else not** *fd__stdin.is__open*;
      stdout__closed: *is__terminated* **implies** *fd__stdout* = *Void* **or else not** *fd__stdout.is__open*;
      stderr__closed: *is__terminated* **implies** *fd__stderr* = *Void* **or else not** *fd__stderr.is__open*;
      terminated: *suspend* **implies** *is__terminated*;

> *pid_invalid*: *is_terminated* **implies not** *is_pid_valid*
feature(*s*) **from** *ABSTRACT_EXEC_PROCESS*
  -- Access
  program_name: *STDC_PATH*
    -- Program to execute
  arguments: *ARRAY*[*STRING*]
    -- Arguments to pass to *program_name*
invariant
  --2007-12-13: invariant failure in some cases, root cause not determined yet
  --pid_known_is_not_terminated: is_pid_valid = not is_terminated
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
  program_name_not_empty: *program_name* /= *Void* **and then not** *program_name.is_empty*;
  arguments_not_void: *arguments* /= *Void*;
  all_arguments_not_void: **not** *has_void_argument*(*arguments*);
  descriptors_are_owners: (*fd_stdin* /= *Void* **and then** *fd_stdin.is_open* **implies** *fd_stdin.is_owner*)
and then (*fd_stdout* /= *Void* **and then** *fd_stdout.is_open* **implies** *fd_stdout.is_owner*)
and then (*fd_stderr* /= *Void* **and then** *fd_stderr.is_open* **implies** *fd_stderr.is_owner*);
end *of* **deferred** *ABSTRACT_EXEC_PROCESS*

B.3   *Short form of ABSTRACT__FILE__DESCRIPTOR*

deferred class *interface ABSTRACT_FILE_DESCRIPTOR*
feature(*s*) **from** *MEMORY*
  dispose
    -- Close handle if owner.
feature(*s*) **from** *KI_INPUT_STREAM*
  -- Input
  non_blocking_read_character
    -- Read the next item in input stream.
    -- Make the result available in *last_item.*
  non_blocking_read_to_buffer (*a_buffer*: *KI_BUFFER*[*CHARACTER*]; *pos, nb*: *INTEGER*):
INTEGER
    -- Fill *a_buffer*, starting at position *pos*, with
    -- at most *nb* items read from input stream.
    -- Return the number of items actually read.
feature(*s*) **from** *KI_INPUT_STREAM*
  -- Status report
  is_closable_for_reading: *BOOLEAN*
    -- Can current input stream be closed?
  is_open_read: *BOOLEAN*
    -- Can items be read from input stream?
  is_rewindable: *BOOLEAN*
    -- Can current input stream be rewound to return input from
    -- the beginning of the stream?
  end_of_input: *BOOLEAN*
    -- Has end-of-file been reached?
  valid_unread_character (*a_character*: *CHARACTER*): *BOOLEAN*
    -- Can *a_character* be put back in input stream?
feature(*s*) **from** *KI_INPUT_STREAM*
  -- Access
  name: *STDC_PATH*
    -- Scratch path
  last_character: *CHARACTER*
    -- Last character read by *read_character* and a few other
    -- routines
feature(*s*) **from** *KI_INPUT_STREAM*
  -- Basic operations
  close_for_reading
    -- Try to close input stream if it is closable. Set
    -- *is_open_read* to false if operation was successful.
  rewind
    -- Move input positionto the beginning of stream.
feature(*s*) **from** *KL_IMPORTED_ANY_ROUTINES*
  -- Access
  *any_*: *KL_ANY_ROUTINES*
    -- Routines that ought to be in class ANY
feature(*s*) **from** *KI_CHARACTER_INPUT_STREAM*
  -- Input

non_blocking_read_string (*nb*: *INTEGER*)
    -- Read at most *nb* characters from input stream.
    -- Make the characters that have actually been read
    -- available in *last_string*.
non_blocking_read_to_string (*a_string*: *STRING*; *pos, nb*: *INTEGER*): *INTEGER*
    -- Fill *a_string*, starting at position *pos*, with
    -- at most *nb* characters read from input stream.
    -- Return the number of characters actually read.
feature(*s*) **from** *KI_CHARACTER_INPUT_STREAM*
  -- Access
last_string: *STRING*
    -- Last string read;
    -- (Note: this query always return the same object.
    -- Therefore a clone should be used if the result
    -- is to be kept beyond the next call to this feature.
    -- However *last_string* is not shared between file objects.)
feature(*s*) **from** *EPX_CHARACTER_INPUT_STREAM*
  -- Access
is_streaming: *BOOLEAN*
    -- Is data coming from a network stream?
feature(*s*) **from** *EPX_CHARACTER_INPUT_STREAM*
  -- Input
last_read: *INTEGER*
    -- Last bytes read by *read_buffer*;
    -- Can be less than requested for non-blocking input.
    -- Check *last_blocked* in that case.
read_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
    -- Read data into *buf* at *offset* for *nbytes* bytes.
    -- Number of bytes actually read are available in *last_read*.
    -- This is a more safe version of *read* in case you have a
    -- STDC_BUFFER object.
feature(*s*) **from** *KI_TEXT_INPUT_STREAM*
  -- Input
read_line
    -- Read characters from input stream until a line separator
    -- or end of file is reached. Make the characters that have
    -- been read available in *last_string* and discard the line
    -- separator characters from the input stream.
    -- Zero characters will be read when non-blocking i/o
    -- is enabled, and *read_line* would block at the first character.
    -- If a character has been read, *read_line* will block until
    -- a %N has been read or *end_of_input* occurs.
read_new_line
    -- Read a line separator from input file.
    -- Make the characters making up the recognized
    -- line separator available in *last_string*,
    -- or make *last_string* empty and leave the
    -- input file unchanged if no line separator
    -- was found.

feature(*s*) **from** *KI_TEXT_INPUT_STREAM*
-- Access
eol: *STRING*
   -- Line separator;
   -- EPX classes do not distinguish between a %R%N or just %N
   -- end-of-line. The platform may though.
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
-- The singleton, available to any because its used in preconditions
security: *STDC_SECURITY*
   -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
-- Access
errno: *STDC_ERRNO*
   -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
-- Status
raise_exception_on_error: *BOOLEAN*
   -- Should an exception be raised when an error occurs?
   -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
-- Change
set_default_action_on_error
   -- Use *security.error_handling.exceptions_enabled* to
   -- determine if an exception should be raised when a C call
   -- returns an error.
set_raise_exception_on_error
   -- Always raise an exception when a C call returns an error.
set_continue_on_error
   -- Never raise an exception when a C call returns an error.
inherit_error_handling (*an_instance*: *STDC_BASE*)
   -- Handle errors like *an_instance*
feature(*s*) **from** *KI_OUTPUT_STREAM*
-- Output
put_character (*c*: *CHARACTER*)
   -- Write a character.
append (*an_input_stream*: *KI_INPUT_STREAM*[*CHARACTER*])
   -- Read items of *an_input_stream* until the end
   -- of input is reached, and write these items to
   -- current output stream.
   -- *append* is safe for non-blocking descriptors.
feature(*s*) **from** *KI_OUTPUT_STREAM*
-- Status report
is_open_write: *BOOLEAN*
   -- Can items be written to output stream?
is_closable_for_writing: *BOOLEAN*
   -- Can current output stream be closed?
feature(*s*) **from** *KI_OUTPUT_STREAM*
-- Basic operations
close_for_writing

  -- Try to close output stream if it is closable. Set
  -- *is_open_write* to false if operation was successful.
 feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
  -- Output
 put_string (*a_string*: *STRING*)
  -- Write *a_string* to output stream.
 put_substring (*a_string*: *STRING*; *s, e*: *INTEGER*)
  -- Write substring of *a_string* between indexes
  -- *s* and *e* to output stream.
 put_integer (*i*: *INTEGER*)
  -- Write decimal representation
  -- of *i* to output stream.
  -- Regexp: 0|(-?[1-9][0-9]*)
 put_boolean (*b*: *BOOLEAN*)
  -- Write "True" to output stream if
  -- *b* is true, "False" otherwise.
 feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
  -- Basic operations
 flush
  -- Flush buffered data to disk.
 feature(*s*) **from** *EPX_CHARACTER_OUTPUT_STREAM*
  -- Output
 last_written: *INTEGER*
  -- How many bytes were written by the last call to a routine;
  -- Can be less than requested for non-blocking output.
  -- Check *last_blocked* in that case.
 put_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
  -- More safe version of *write* in case you have a
  -- STDC_BUFFER object.
 write_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
  -- More safe version of *write* in case you have a
  -- STDC_BUFFER object.
 feature(*s*) **from** *KI_TEXT_OUTPUT_STREAM*
  -- Output
 put_line (*a_string*: *STRING*)
  -- Write *a_string* to output stream
  -- followed by a line separator.
 put_new_line
  -- Write a line separator to output stream.
 feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*
  -- Basic operations
 close
  -- Close the resource.
 feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*
  -- Status report
 is_closable: *BOOLEAN*
  -- Can current stream be closed for reading and writing?
 is_open: *BOOLEAN*
  -- Does *handle* contain an open handle?

is_owner: *BOOLEAN*
    -- Does this object close the stream on *close* or *dispose*?
    -- Only for resources that are owned, are resource limits checked.
feature(*s*) **from** *STDC_HANDLE*
  -- Access
resource_usage_can_be_increased: *BOOLEAN*
    -- Is it allowed to open another file?
feature(*s*) **from** *STDC_HANDLE*
  -- Influence ownership of the handle. Can help to influence subtle garbage collector problems
become_owner
    -- This class will own its handle. This is the only function
    -- that actually increases the resource count.
unown
    -- Resource will not be closed on dispose. Calling close will
    -- be forbidden. This routine may not call any other object,
    -- else it cannot be called from within dispose.
feature(*s*) **from** *STDC_HANDLE*
  -- Close
detach
    -- Forget the resource. Resource is not closed.
    -- You cannot read and write anymore.
feature(*s*) **from** *STDC_HANDLE*
  -- Resource
capacity: *INTEGER*
    -- Number of resources that are in use by *handle*. For a
    -- file this is 1, for a memory handle, this is the number of
    -- bytes.
fd: *H*
    -- Identifier of resource tracked by this class.
feature(*s*) **from** *PORTABLE_PATH*
  -- Change
set_portable_path (*a_path*: *STRING*)
    -- Set *portable_path* to *a_path*.
feature(*s*) **from** *HASHABLE*
hash_code: *INTEGER*
    -- Hash code value
feature(*s*) **from** *STDC_HANDLE_BASED_IO_STREAM*
  -- Stream or disk file
set_streaming (*enable*: *BOOLEAN*)
    -- Influence behaviour of certain functions if they should be
    -- optimized for data coming from disk or data coming from
    -- the network. In particular *is_streaming* implies that a
    -- client application is prepared to handle *read*s that
    -- return less than the requested number of bytes, but dont
    -- assume that means end-of-file.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Initialization
make
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*

-- Special creation

attach_to_fd (*a_fd*: *INTEGER*; *a_become_owner*: *BOOLEAN*)
  -- Create descriptor with value *a_fd*. Descriptor will close
  -- it when *a_become_owner*.

make_as_duplicate (*another*: *ABSTRACT_DESCRIPTOR*)
  -- On creation, create a duplicate from another descriptor.
  -- As normal call, closes its own descriptor first (if open) and
  -- duplicates next.

feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Read and write to memory block

last_blocked: *BOOLEAN*
  -- Would last call to *read* or *write* block?

read (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
  -- Read data into *buf* at *offset* for *nbytes* bytes.
  -- The number of bytes actually read, is available in *last_read*.

write (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
  -- Write given data from *buf* at *offset*, for *nbytes*
  -- bytes. Number of actually written bytes are in
  -- *last_written*. *last_written* can be unequal to *nbytes*
  -- if i/o is non-blocking or some error has occurred.

feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Eiffel like output

put (*a*: *ANY*)
  -- Write any Eiffel object as string using its *out* value.

write_character (*c*: *CHARACTER*)
  -- Write a character.

write_string (*a_string*: *STRING*)
  -- Write *a_string* to output stream.

puts (*a_string*: *STRING*)
  -- Write *a_string* to output stream.

feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Buffered input

read_character
  -- Sets *last_character*.
  -- If this routine blocks, *last_character* has the value
  -- %U. Therefore, if non-blocking is enabled, always check
  -- *last_blocked* to see if the value make sense.

read_string (*nb*: *INTEGER*)
  -- Read at most *nb* characters from input stream.
  -- Make the characters that have actually been read
  -- available in *last_string*.
  -- Zero characters will be read when non-blocking i/o
  -- is enabled, and *read* would block.

feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Status report

is_attached_to_terminal: *BOOLEAN*
  -- Is the handle associated with character device?

feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Access

value: *INTEGER*
   -- The actual file descriptor value
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- non-blocking i/o
  is_blocking_io: *BOOLEAN*
    -- Is blocking i/o enabled?
    -- Blocking i/o is the default.
    -- If false, calls like *read* and *write* will never wait
    -- for input, if there is no input.
  set_blocking_io (*enable*: *BOOLEAN*)
    -- Set *is_blocking_io*.
  supports_nonblocking_io: *BOOLEAN*
    -- Does this descriptor support non-blocking input/output?
    -- On POSIX systems, any descriptor does.
    -- On Windows, sockets and pipes do.
feature(*s*) **from** *ABSTRACT_FILE_DESCRIPTOR*
  -- Initialization
  open (*a_path*: *STRING*; *a_flags*: *INTEGER*)
    -- Open given file with access given by *flags*.
  open_read (*a_path*: *STRING*)
    -- Open given file with access given by *flags*.
  open_write (*a_path*: *STRING*)
  open_read_write (*a_path*: *STRING*)
  open_truncate (*a_path*: *STRING*)
    -- Open file, if it exists, truncate it first.
  create_read_write (*a_path*: *STRING*)
    -- Always create a file, existing or not.
    -- Give read/write permissions to user only.
  create_write (*a_path*: *STRING*)
    -- Always create a file, existing or not.
    -- Give read/write permissions to user only.
  create_with_mode (*a_path*: *STRING*; *flags, mode*: *INTEGER*)
    -- Create a file according to *flags* and with *mode* access
    -- permissions. Make sure you have th O_CREAT flag in flags
    -- if you really want to create something!
feature(*s*) **from** *ABSTRACT_FILE_DESCRIPTOR*
  -- File position
  seek (*offset*: *INTEGER*)
    -- Set file position to given absolute *offset*.
  seek_from_current (*offset*: *INTEGER*)
    -- Set file position relative to current position.
  seek_from_end (*offset*: *INTEGER*)
    -- Set file position relative to end of file.
feature(*s*) **from** *ABSTRACT_FILE_DESCRIPTOR*
  -- Access
  status: *EPX_STATUS*
    -- The status for this file descriptor;
    -- Value is cached, recreated only when file reopened.
    -- Call *status.refresh* to get updated values.

invariant

    accessing_real_singleton: *security_is_real_singleton*;

    valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;

    open_in_sync: *is_open_read* **or** *is_open_write* **implies** *is_open*; -- The reverse is not true, for examples sockets can be
 -- closed for reading/writing, but still open.

    capacity_not_negative: *capacity* $>= 0$;

    valid_capacity: *is_open* $=$ (*capacity* > 0);

    open_implies_handle_assigned: *is_open* $=$ (*fd* $/=$ *unassigned_value*);

    owned_implies_open: *is_owner* **implies** *is_open*;

    owned_implies_handle_assigned: *is_owner* **implies** *fd* $/=$ *unassigned_value*;

    line_buffer_index_offset_ok: *line_buffer* $/=$ *Void* **implies** *line_buffer_index* $<=$ *line_buffer.count*;

    valid_status: **not** *is_open* **implies** *my_status* $=$ *Void*;

end *of* **deferred** *ABSTRACT_FILE_DESCRIPTOR*

B.4 Short form of ABSTRACT_FILE_SYSTEM

deferred class *interface ABSTRACT_FILE_SYSTEM*
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
   -- The singleton, available to any because its used in preconditions
   security: *STDC_SECURITY*
     -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
   -- Access
   errno: *STDC_ERRNO*
     -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
   -- Status
   raise_exception_on_error: *BOOLEAN*
     -- Should an exception be raised when an error occurs?
     -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
   -- Change
   set_default_action_on_error
     -- Use *security.error_handling.exceptions_enabled* to
     -- determine if an exception should be raised when a C call
     -- returns an error.
   set_raise_exception_on_error
     -- Always raise an exception when a C call returns an error.
   set_continue_on_error
     -- Never raise an exception when a C call returns an error.
   inherit_error_handling (*an_instance*: *STDC_BASE*)
     -- Handle errors like *an_instance*
feature(*s*) **from** *STDC_FILE_SYSTEM*
   -- Path names
   expand_path (*a_path*: *STRING*): *STDC_PATH*
     -- returns a new path
feature(*s*) **from** *STDC_FILE_SYSTEM*
   -- Rename files/directories, remove files/directories
   remove_file (*a_path*: *STRING*)
     -- Removes a file from a directory.
     -- For Standard C, its implementation defined what
     -- remove_file does if file is opened by some process
     -- (*remove_file* fails on Windows for example).
     -- doesnt remove a directory.
   rename_to (*current_path, new_path*: *STRING*)
     -- Rename a file or a directory.
     -- *new_path* should not be an existing path.
feature(*s*) **from** *STDC_FILE_SYSTEM*
   -- Accessibility of files
   is_modifiable (*a_path*: *STRING*): *BOOLEAN*
     -- tests if file is readable and writable by this program
     -- uses real user ID and real group ID instead of effective ones
   is_readable (*a_path*: *STRING*): *BOOLEAN*

        -- Tests if *a_path* is readable by this program. *a_path*
        -- can be a file or a directory.
        -- Uses real user ID and real group ID instead of effective
        -- ones.
  feature(*s*) **from** *STDC_FILE_SYSTEM*
    -- File and string
    write_string_to_file (*s, a_file_name*: *STRING*)
      -- Write *s* to file *a_file_name*.
  feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
    -- Directory access
    change_directory (*a_directory*: *STRING*)
      -- Changes the current working directory.
    current_directory: *STRING*
      -- The current directory
    make_directory (*a_directory*: *STRING*)
      -- Makes a directory, only accessible by owner.
    mkdir (*a_directory*: *STRING*)
      -- Makes a directory, only accessible by owner.
    remove_directory (*a_directory*: *STRING*)
      -- Removes an empty directory. See also *force_remove_directory*.
    rmdir (*a_directory*: *STRING*)
      -- Removes an empty directory. See also *force_remove_directory*.
    force_remove_directory (*a_directory*: *STRING*)
      -- Removes a directory, even when not empty.
      -- I suggest you do not have hard or symbolic links in *a_directory*...
  feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
    -- File statistics
    status (*a_path*: *STRING*): *ABSTRACT_STATUS_PATH*
      -- Get information about a file.
      require
        valid_path: *a_path* /= *Void* **and then not** *a_path.is_empty*;
        existing_file: *is_existing*(*a_path*)
      ensure
        status_returned: *Result* /= *Void*
    status_may_fail (*a_path*: *STRING*): *ABSTRACT_STATUS_PATH*
      -- Retrieve status information for *a_path*. *a_path* may or
      -- may not exist. Check *Result.found* to see if statistics
      -- were retrieved.
      require
        valid_path: *a_path* /= *Void* **and then not** *a_path.is_empty*
      ensure
        status_returned: *Result* /= *Void*
  feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
    -- Directory browsing
    browse_directory (*a_path*: *STRING*): *EPX_DIRECTORY*
      -- Get information about a directory.
      require
        valid_path: *a_path* /= *Void* **and then not** *a_path.is_empty*;
        path_is_directory: *status_may_fail*(*a_path*).*found* **and then** *status_may_fail*(*a_path*).*is_directory*

     ensure
       directory_returned: *Result* /= *Void*
   find_program_in_path (*a_filename*: *STRING*; *a_paths*: *ARRAY*[*STRING*]): *STRING*
    -- Look for *a_filename* in *a_paths*, check if it is a
    -- binary and return the full path to *a_filename* when
    -- found. Return Void if not found.
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
  -- Accessibility of files
  last_access_result: *INTEGER*
    -- value of last access test
  is_accessible (*a_path*: *STRING*; *a_mode*: *INTEGER*): *BOOLEAN*
    -- Is *a_path* accessibility using *a_mode*?
  access (*a_path*: *STRING*; *a_mode*: *INTEGER*): *BOOLEAN*
    -- Is *a_path* accessibility using *a_mode*?
  is_directory (*a_path*: *STRING*): *BOOLEAN*
    -- Does *a_path* exists and is it a directory?
  is_existing (*a_path*: *STRING*): *BOOLEAN*
    -- Is *a_path* an existing file, directory, whatever?
    -- Tests if file does exist, not if it is readable or writable by
    -- this program!
    -- Uses real user ID and real group ID instead of effective ones.
  is_empty (*a_path*: *STRING*): *BOOLEAN*
    -- True if file exists and has a size equal to zero.
  is_executable (*a_path*: *STRING*): *BOOLEAN*
    -- tests if file is executable by this program
  is_regular_file (*a_path*: *STRING*): *BOOLEAN*
    -- Does *a_path* exists and is it a regular file?
  is_writable (*a_path*: *STRING*): *BOOLEAN*
    -- tests if file is writable by this program
    -- uses real user ID and real group ID instead of effective ones
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
  -- File system properties
  is_case_sensitive: *BOOLEAN*
    -- is file system case sensitive or not?
    -- This query is dedicated to jwz
  path_separator: *CHARACTER*
    -- What is the path separator?
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
  -- File and string
  file_content_as_string (*a_file_name*: *STRING*): *STRING*
    -- Contents of *a_file_name* as a STRING
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
  -- Path names
  resolved_path_name (*a_path*: *STRING*): *STRING*
    -- Absolute pathname derived from *a_path* that names the
    -- same file, whose resolution does not involve ".", "..", or
    -- symbolic links
  temporary_directory: *STRING*
    -- The name of the temporary directory;

-- Name does not end with the directory separator.
    ensure
        directory_returned: *Result* /= *Void*;
        directory_exists: *is_directory*(*Result*);
        directory_is_writable: *is_modifiable*(*Result*);
        last_char_not_separator: *Result.item*(*Result.count*) /= *path_separator*
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
end *of* **deferred** *ABSTRACT_FILE_SYSTEM*

B.5 Short form of ABSTRACT_HOST

deferred class *interface ABSTRACT_HOST*
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
   -- The singleton, available to any because its used in preconditions
   security: *STDC_SECURITY*
      -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
   -- Access
   errno: *STDC_ERRNO*
      -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
   -- Status
   raise_exception_on_error: *BOOLEAN*
      -- Should an exception be raised when an error occurs?
      -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
   -- Change
   set_default_action_on_error
      -- Use *security.error_handling.exceptions_enabled* to
      -- determine if an exception should be raised when a C call
      -- returns an error.
   set_raise_exception_on_error
      -- Always raise an exception when a C call returns an error.
   set_continue_on_error
      -- Never raise an exception when a C call returns an error.
   inherit_error_handling (*an_instance*: *STDC_BASE*)
      -- Handle errors like *an_instance*
feature(*s*) **from** *ABSTRACT_HOST*
   -- Initialization
   make_from_name (*a_name*: *STRING*)
      -- Initialize host from *name*. If *name* is numerical, the
      -- behaviour is not specified.
   make_from_address (*an_address*: *ABSTRACT_IP_ADDRESS*)
      -- Initialize host from ip address *an_address*.
      -- An attempt is made to resolve the host name using this address.
      -- Status is always found, even when reverse lookup failed.
   make_from_ip4_any
      -- IP address that refers to all local interfaces.
   make_from_ip4_loopback
      -- IP address that refers to the loopback device.
      -- No attempt at resolving is done.
feature(*s*) **from** *ABSTRACT_HOST*
   -- Command
   find_by_address
      -- Attempt to lookup up the host by first ip address in
      -- *addresses*. Sets *found* if host could be found.
      -- If found, sets *canonical_name*, *aliases*,
      -- *address_family*, *address_length* and *addresses*.

_find_by_name_
    -- Attempt to lookup up the host given in _name_. Sets
    -- _found_ if host could be found.
    -- If found, sets _canonical_name, aliases,_
    -- _address_family, address_length_ and _addresses._
feature(_s_) **from** _ABSTRACT_HOST_
  -- Status
found: _BOOLEAN_
    -- Does this class contain a resolved host?
    -- If False, _not_found_reason_ contains the reason.
feature(_s_) **from** _ABSTRACT_HOST_
  -- Access
name: _STRING_
    -- Name as given to _make_from_name_ or else equal to
    -- _canonical_name_
_not_found_reason_: _INTEGER_
    -- Reason why _found_ is False;
    -- The interpretation of this value depends on the platform.
_canonical_name_: _STRING_
    -- Official (canonical) name of host.
aliases: _ARRAY[STRING]_
    -- Alias names.
_address_family_: _INTEGER_
    -- Host address type: AF_INET or AF_INET6
_address_length_: _INTEGER_
    -- Length of address: 4 or 16.
addresses: _ARRAY[ABSTRACT_IP_ADDRESS]_
    -- Array with IPv4 or IPv6 addresses.
invariant
  _accessing_real_singleton_: _security_is_real_singleton_;
  _valid_error_action_: _error_action_ $>=$ _0_ **and** _error_action_ $<=$ _2_;
  _name_void_or_not_empty_: _name_ $=$ _Void_ **or else not** _name.is_empty_;
  _has_canonical_name_: _found_ **implies** _name_ $/=$ _Void_ $=$ (_canonical_name_ $/=$ _Void_);
  _has_at_least_one_ip_address_: _found_ $=$ (_addresses_ $/=$ _Void_ **and then** _addresses.count_
> 0);
  _only_non_void_addresses_: _found_ **implies** _is_every_address_not_void_;
  _has_aliases_: _found_ $=$ (_aliases_ $/=$ _Void_);
  _valid_length_: _found_ **implies** _address_length_ > 0;
  consistent: _addresses_ $/=$ _Void_ **and then** _addresses.count_ > 0 **implies** _found_;
  _my_not_found_reason_valid_: _found_ $=$ (_my_not_found_reason_ $=$ _0_);
end _of_ **deferred** _ABSTRACT_HOST_

B.6 Short form of ABSTRACT__IP4__ADDRESS

class *interface ABSTRACT__IP4__ADDRESS*
feature(*s*) **from** *STDC__SECURITY__ACCESSOR*
   -- The singleton, available to any because its used in preconditions
   security: *STDC__SECURITY*
     -- Singleton entry point for security.
feature(*s*) **from** *STDC__BASE*
   -- Access
   errno: *STDC__ERRNO*
     -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC__BASE*
   -- Status
   raise__exception__on__error: *BOOLEAN*
     -- Should an exception be raised when an error occurs?
     -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC__BASE*
   -- Change
   set__default__action__on__error
     -- Use *security.error__handling.exceptions__enabled* to
     -- determine if an exception should be raised when a C call
     -- returns an error.
   set__raise__exception__on__error
     -- Always raise an exception when a C call returns an error.
   set__continue__on__error
     -- Never raise an exception when a C call returns an error.
   inherit__error__handling (*an__instance*: *STDC__BASE*)
     -- Handle errors like *an__instance*
feature(*s*) **from** *ABSTRACT__IP__ADDRESS*
   -- Initialization
   make__from__pointer (*a__ptr*: *POINTER*)
     -- Initialize ip address from 32-bit integer pointed to by *a__ptr*.
     -- We assume *a__ptr* points to a value in network byte order.
feature(*s*) **from** *ABSTRACT__IP__ADDRESS*
   -- Status
   is__loopback__address: *BOOLEAN*
     -- Does this IP address refer to the loopback address?
feature(*s*) **from** *ABSTRACT__IP__ADDRESS*
   -- General ip address features
   address__family: *INTEGER*
     -- Is it an ip4 or ip6 address.
   address__length: *INTEGER*
     -- Length of an IPv4 address is 4.
   ptr: *POINTER*
     -- Pointer to an in__addr or in6__addr structure.
     -- (bytes are in network byte order for in__addr)
feature(*s*) **from** *ABSTRACT__IP4__ADDRESS*
   -- Initialization
   make__from__any

    -- Initialize using the any address (i.e. 0.0.0.0).

  make_from_integer (*a_value*: *INTEGER*)

    -- Initialize ip address from 32-bit integer.

  make_from_loopback

    -- Initialize using the loopback address (i.e. 127.0.0.1).

  make_from_components (*a1, a2, a3, a4*: *INTEGER*)

    -- Make IP4 address given the four individual fields of an IP

    -- 4 address.

feature(*s*) **from** *ABSTRACT_IP4_ADDRESS*

  -- Access

  value: *INTEGER*

    -- IPv4 address as 32-bit integer.

    -- Value is in host byte order.

feature(*s*) **from** *ABSTRACT_IP4_ADDRESS*

  -- Change

  set_value (*new_value*: *INTEGER*)

    -- Change IP address *value* to *new_value.*

feature(*s*) **from** *ABSTRACT_IP4_ADDRESS*

  -- Comparison

  is_equal (*other*: **like** *Current*): *BOOLEAN*

    -- Is *other* IP4 address equal to this IP address?

feature(*s*) **from** *ABSTRACT_IP4_ADDRESS*

  -- Output

  out: *STRING*

    -- Friendly out

invariant

  accessing_real_singleton: *security_is_real_singleton*;

  valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;

  buf_not_void: *buf* $/=$ *Void*;

  buf_capacity_large_enough: *buf.capacity* $>=$ *abstract_api.posix_in_addr_size*;

end *of ABSTRACT_IP4_ADDRESS*

B.7 Short form of ABSTRACT_IP6_ADDRESS

deferred class *interface ABSTRACT_IP6_ADDRESS*
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
   -- The singleton, available to any because its used in preconditions
   security: *STDC_SECURITY*
      -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
   -- Access
   errno: *STDC_ERRNO*
      -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
   -- Status
   raise_exception_on_error: *BOOLEAN*
      -- Should an exception be raised when an error occurs?
      -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
   -- Change
   set_default_action_on_error
      -- Use *security.error_handling.exceptions_enabled* to
      -- determine if an exception should be raised when a C call
      -- returns an error.
   set_raise_exception_on_error
      -- Always raise an exception when a C call returns an error.
   set_continue_on_error
      -- Never raise an exception when a C call returns an error.
   inherit_error_handling (*an_instance*: *STDC_BASE*)
      -- Handle errors like *an_instance*
feature(*s*) **from** *ABSTRACT_IP_ADDRESS*
   -- Initialization
   make_from_pointer (*a_ptr*: *POINTER*)
      -- Initialize ip address from 32-bit integer.
feature(*s*) **from** *ABSTRACT_IP_ADDRESS*
   -- Status
   is_loopback_address: *BOOLEAN*
      -- Does this IP address refer to the loopback address?
feature(*s*) **from** *ABSTRACT_IP_ADDRESS*
   -- General ip address features
   address_family: *INTEGER*
      -- Is it an ip4 or ip6 address.
   address_length: *INTEGER*
      -- Length of an IPv6 address is 16.
   ptr: *POINTER*
      -- Pointer to an in_addr or in6_addr structure.
      -- (bytes are in network byte order for in_addr)
feature(*s*) **from** *ABSTRACT_IP6_ADDRESS*
   -- Comparison
   is_equal (*other*: **like** *Current*): *BOOLEAN*
      -- Is *other* IP4 address equal to this IP address?

feature(*s*) **from** *ABSTRACT_IP6_ADDRESS*
   -- Output
   out: *STRING*
     -- Friendly out
feature(*s*) **from** *ABSTRACT_IP6_ADDRESS*
   -- General ip address features
   scope_id: *INTEGER*
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
   buf_not_void: *buf* $/=$ *Void*;
   buf_capacity_large_enough: *buf.capacity* $>=$ *abstract_api.posix_in6_addr_size*;
end *of* **deferred** *ABSTRACT_IP6_ADDRESS*

B.8 Short form of ABSTRACT__PIPE

class *interface ABSTRACT__PIPE*
feature(*s*) **from** *ABSTRACT__PIPE*
  -- Pipe operations
  close
feature(*s*) **from** *ABSTRACT__PIPE*
  -- Access
  fdout: *ABSTRACT__FILE__DESCRIPTOR*
    -- Outgoing end of pipe
  fdin: *ABSTRACT__FILE__DESCRIPTOR*
    -- Incoming end of pipe
invariant
  accessing__real__singleton: *security__is__real__singleton*;
  valid__error__action: *error__action* >= *0* **and** *error__action* <= *2*;
  valid__pipe: *fdin* /= *Void* **and** *fdout* /= *Void*;
end *of ABSTRACT__PIPE*

B.9 Short form of ABSTRACT_SERVICE

deferred class *interface ABSTRACT_SERVICE*
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
    -- The singleton, available to any because its used in preconditions
    security: *STDC_SECURITY*
        -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
    -- Access
    errno: *STDC_ERRNO*
        -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
    -- Status
    raise_exception_on_error: *BOOLEAN*
        -- Should an exception be raised when an error occurs?
        -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
    -- Change
    set_default_action_on_error
        -- Use *security.error_handling.exceptions_enabled* to
        -- determine if an exception should be raised when a C call
        -- returns an error.
    set_raise_exception_on_error
        -- Always raise an exception when a C call returns an error.
    set_continue_on_error
        -- Never raise an exception when a C call returns an error.
    inherit_error_handling (*an_instance*: *STDC_BASE*)
        -- Handle errors like *an_instance*
feature(*s*) **from** *ABSTRACT_SERVICE*
    -- Initialization
    make_from_name (*a_name, a_protocol*: *STRING*)
        -- Retrieve service information with *a_name* and optional
        -- *a_protocol* from services database.
        -- If service not found, an exception is raised.
    make_from_name_with_default (*a_name, a_protocol*: *STRING*; *a_default_port*: *INTEGER*)
        -- Retrieve service information with *a_name* and optional
        -- *a_protocol* from services database.
        -- If service not found, *a_default_port* is used for *port*.
    make_from_ephemeral_port (*a_protocol*: *STRING*)
        -- Initialize service, but let kernel choose a port at bind time.
        -- Provide a *a_protocol* if necessary.
    make_from_port (*a_port*: *INTEGER*; *a_protocol*: *STRING*)
        -- Initialize service from given a_port.
        -- Make sure to provide a *a_protocol* if necessary!
feature(*s*) **from** *ABSTRACT_SERVICE*
    -- Access
    port: *INTEGER*
        -- port number if not zero
    name: *STRING*

    -- official service name
   aliases: *ARRAY*[*STRING*]
    -- alias list
   protocol: *STRING*
    -- protocol to use (udp/tcp)
   protocol_type: *INTEGER*
    -- SOCK_STREAM or SOCK_DGRAM
 feature(*s*) **from** *ABSTRACT_SERVICE*
   -- Status
   is_tcp: *BOOLEAN*
    -- Is *protocol_type* the tcp protocl?
   is_udp: *BOOLEAN*
    -- Is *protocol_type* the datagram protocl?
 invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
   name_void_or_not_empty: *name* = *Void* **or else** **not** *name.is_empty*;
   valid_port: *port* >= *0* **and** *port* <= *65535*;
   valid_protocol: *protocol* = *Void* **or else** *protocol.is_empty* **or else** (*protocol.is_equal*(*once_tcp*)
 or *protocol.is_equal*(*once_udp*));
   valid_protocol_type: *protocol_type* = *sock_stream* **or else** *protocol_type* = *sock_dgram*;
   valid_aliases: *aliases* /= *Void*;
 end *of* **deferred** *ABSTRACT_SERVICE*

B.10 Short form of ABSTRACT_STATUS

deferred class *interface ABSTRACT_STATUS*
feature(*s*) **from** *ABSTRACT_STATUS*
  -- Status
  is_open: *BOOLEAN*
    -- Can status be refreshed?
feature(*s*) **from** *ABSTRACT_STATUS*
  -- Change
  refresh
    -- refresh the cached information
    require
      open: *is_open*
feature(*s*) **from** *ABSTRACT_STATUS*
  -- stat members
  atime: *INTEGER*
    -- Unix time of last access.
  access_time: *INTEGER*
    -- Unix time of last access.
  device_number: *INTEGER*
    -- ID of device containing the file.
    -- Windows: Drive number of the disk containing the file.
  is_character_special: *BOOLEAN*
    -- Is this file a character-special file?
  is_directory: *BOOLEAN*
  is_fifo: *BOOLEAN*
  is_regular_file: *BOOLEAN*
  mtime: *INTEGER*
    -- Unix time of last data modification.
  modification_time: *INTEGER*
    -- Unix time of last data modification.
  nlink: *INTEGER*
  number_of_hard_links: *INTEGER*
  size: *INTEGER*
    -- Size of file in bytes.
  status_change_time: *INTEGER*
    -- Unix time of last status change.
    -- For example changing the permission bits will set this time.
feature(*s*) **from** *ABSTRACT_STATUS*
  -- Direct access to the individual stat fields, not recommended
  unix_mode: *INTEGER*
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
  stat_not_void: *stat* $/=$ *Void* **and then** *stat.capacity* $>=$ *abstract_stat_size*;
end *of* **deferred** *ABSTRACT_STATUS*

B.11 Short form of ABSTRACT_TCP_CLIENT_SOCKET

deferred class *interface ABSTRACT_TCP_CLIENT_SOCKET*
feature(*s*) **from** *MEMORY*
   dispose
     -- Close handle if owner.
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Input
   non_blocking_read_character
     -- Read the next item in input stream.
     -- Make the result available in *last_item*.
   non_blocking_read_to_buffer (*a_buffer*: *KI_BUFFER*[*CHARACTER*]; *pos, nb*: *INTEGER*):
INTEGER
     -- Fill *a_buffer*, starting at position *pos*, with
     -- at most *nb* items read from input stream.
     -- Return the number of items actually read.
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Status report
   is_closable_for_reading: *BOOLEAN*
     -- Can current input stream be closed?
   is_open_read: *BOOLEAN*
     -- Can items be read from input stream?
   is_rewindable: *BOOLEAN*
     -- Can current input stream be rewound to return input from
     -- the beginning of the stream?
   end_of_input: *BOOLEAN*
     -- Has end-of-file been reached?
   valid_unread_character (*a_character*: *CHARACTER*): *BOOLEAN*
     -- Can *a_character* be put back in input stream?
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Access
   name: *STDC_PATH*
     -- Scratch path
   last_character: *CHARACTER*
     -- Last character read by *read_character* and a few other
     -- routines
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Basic operations
   close_for_reading
     -- Try to close input stream if it is closable. Set
     -- *is_open_read* to false if operation was successful.
   rewind
     -- Move input position to the beginning of stream.
feature(*s*) **from** *KL_IMPORTED_ANY_ROUTINES*
   -- Access
   *any_*: *KL_ANY_ROUTINES*
     -- Routines that ought to be in class ANY
feature(*s*) **from** *KI_CHARACTER_INPUT_STREAM*
   -- Input

non_blocking_read_string (*nb*: *INTEGER*)
    -- Read at most *nb* characters from input stream.
    -- Make the characters that have actually been read
    -- available in *last_string*.
non_blocking_read_to_string (*a_string*: *STRING*; *pos, nb*: *INTEGER*): *INTEGER*
    -- Fill *a_string*, starting at position *pos*, with
    -- at most *nb* characters read from input stream.
    -- Return the number of characters actually read.
feature(*s*) **from** *KI_CHARACTER_INPUT_STREAM*
  -- Access
last_string: *STRING*
    -- Last string read;
    -- (Note: this query always return the same object.
    -- Therefore a clone should be used if the result
    -- is to be kept beyond the next call to this feature.
    -- However *last_string* is not shared between file objects.)
feature(*s*) **from** *EPX_CHARACTER_INPUT_STREAM*
  -- Access
is_streaming: *BOOLEAN*
    -- Is data coming from a network stream?
feature(*s*) **from** *EPX_CHARACTER_INPUT_STREAM*
  -- Input
last_read: *INTEGER*
    -- Last bytes read by *read_buffer*;
    -- Can be less than requested for non-blocking input.
    -- Check *last_blocked* in that case.
read_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
    -- Read data into *buf* at *offset* for *nbytes* bytes.
    -- Number of bytes actually read are available in *last_read*.
    -- This is a more safe version of *read* in case you have a
    -- STDC_BUFFER object.
feature(*s*) **from** *KI_TEXT_INPUT_STREAM*
  -- Input
read_line
    -- Read characters from input stream until a line separator
    -- or end of file is reached. Make the characters that have
    -- been read available in *last_string* and discard the line
    -- separator characters from the input stream.
    -- Zero characters will be read when non-blocking i/o
    -- is enabled, and *read_line* would block at the first character.
    -- If a character has been read, *read_line* will block until
    -- a %N has been read or *end_of_input* occurs.
read_new_line
    -- Read a line separator from input file.
    -- Make the characters making up the recognized
    -- line separator available in *last_string*,
    -- or make *last_string* empty and leave the
    -- input file unchanged if no line separator
    -- was found.

feature(*s*) **from** *KI__TEXT__INPUT__STREAM*
   -- Access
   eol: *STRING*
     -- Line separator;
     -- EPX classes do not distinguish between a %R%N or just %N
     -- end-of-line. The platform may though.
feature(*s*) **from** *STDC__SECURITY__ACCESSOR*
   -- The singleton, available to any because its used in preconditions
   security: *STDC__SECURITY*
     -- Singleton entry point for security.
feature(*s*) **from** *STDC__BASE*
   -- Access
   errno: *STDC__ERRNO*
     -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC__BASE*
   -- Status
   raise__exception__on__error: *BOOLEAN*
     -- Should an exception be raised when an error occurs?
     -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC__BASE*
   -- Change
   set__default__action__on__error
     -- Use *security.error__handling.exceptions__enabled* to
     -- determine if an exception should be raised when a C call
     -- returns an error.
   set__raise__exception__on__error
     -- Always raise an exception when a C call returns an error.
   set__continue__on__error
     -- Never raise an exception when a C call returns an error.
   inherit__error__handling (*an__instance*: *STDC__BASE*)
     -- Handle errors like *an__instance*
feature(*s*) **from** *KI__OUTPUT__STREAM*
   -- Output
   put__character (*c*: *CHARACTER*)
     -- Write a character.
   append (*an__input__stream*: *KI__INPUT__STREAM*[*CHARACTER*])
     -- Read items of *an__input__stream* until the end
     -- of input is reached, and write these items to
     -- current output stream.
     -- *append* is safe for non-blocking descriptors.
feature(*s*) **from** *KI__OUTPUT__STREAM*
   -- Status report
   is__open__write: *BOOLEAN*
     -- Can items be written to output stream?
   is__closable__for__writing: *BOOLEAN*
     -- Can current output stream be closed?
feature(*s*) **from** *KI__OUTPUT__STREAM*
   -- Basic operations
   close__for__writing

      -- Try to close output stream if it is closable. Set
      -- *is_open_write* to false if operation was successful.
  feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
    -- Output
    put_string (*a_string*: *STRING*)
      -- Write *a_string* to output stream.
    put_substring (*a_string*: *STRING*; *s, e*: *INTEGER*)
      -- Write substring of *a_string* between indexes
      -- *s* and *e* to output stream.
    put_integer (*i*: *INTEGER*)
      -- Write decimal representation
      -- of *i* to output stream.
      -- Regexp: 0|(-?[1-9][0-9]*)
    put_boolean (*b*: *BOOLEAN*)
      -- Write "True" to output stream if
      -- *b* is true, "False" otherwise.
  feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
    -- Basic operations
    flush
      -- Flush buffered data to disk.
  feature(*s*) **from** *EPX_CHARACTER_OUTPUT_STREAM*
    -- Output
    last_written: *INTEGER*
      -- How many bytes were written by the last call to a routine;
      -- Can be less than requested for non-blocking output.
      -- Check *last_blocked* in that case.
    put_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
      -- More safe version of *write* in case you have a
      -- STDC_BUFFER object.
    write_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
      -- More safe version of *write* in case you have a
      -- STDC_BUFFER object.
  feature(*s*) **from** *KI_TEXT_OUTPUT_STREAM*
    -- Output
    put_line (*a_string*: *STRING*)
      -- Write *a_string* to output stream
      -- followed by a line separator.
    put_new_line
      -- Write a line separator to output stream.
  feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*
    -- Basic operations
    close
      -- Close the resource.
  feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*
    -- Status report
    is_closable: *BOOLEAN*
      -- Can current stream be closed for reading and writing?
    is_open: *BOOLEAN*
      -- Does *handle* contain an open handle?

is_owner: *BOOLEAN*
    -- Does this object close the stream on *close* or *dispose*?
    -- Only for resources that are owned, are resource limits checked.
feature(*s*) **from** *STDC_HANDLE*
  -- Access
resource_usage_can_be_increased: *BOOLEAN*
    -- Is it allowed to open another file?
feature(*s*) **from** *STDC_HANDLE*
  -- Influence ownership of the handle. Can help to influence subtle garbage collector problems
become_owner
    -- This class will own its handle. This is the only function
    -- that actually increases the resource count.
unown
    -- Resource will not be closed on dispose. Calling close will
    -- be forbidden. This routine may not call any other object,
    -- else it cannot be called from within dispose.
feature(*s*) **from** *STDC_HANDLE*
  -- Close
detach
    -- Forget the resource. Resource is not closed.
    -- You cannot read and write anymore.
feature(*s*) **from** *STDC_HANDLE*
  -- Resource
capacity: *INTEGER*
    -- Number of resources that are in use by *handle*. For a
    -- file this is 1, for a memory handle, this is the number of
    -- bytes.
fd: *H*
    -- Identifier of resource tracked by this class.
feature(*s*) **from** *PORTABLE_PATH*
  -- Change
set_portable_path (*a_path*: *STRING*)
    -- Set *portable_path* to *a_path*.
feature(*s*) **from** *HASHABLE*
hash_code: *INTEGER*
    -- Hash code value
feature(*s*) **from** *STDC_HANDLE_BASED_IO_STREAM*
  -- Stream or disk file
set_streaming (*enable*: *BOOLEAN*)
    -- Influence behaviour of certain functions if they should be
    -- optimized for data coming from disk or data coming from
    -- the network. In particular *is_streaming* implies that a
    -- client application is prepared to handle *read*s that
    -- return less than the requested number of bytes, but dont
    -- assume that means end-of-file.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Initialization
make
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*

    -- Special creation
    attach_to_socket (*a_fd*: *INTEGER*; *a_become_owner*: *BOOLEAN*)
       -- Create descriptor with value *a_fd*. Descriptor will close
       -- it when *a_become_owner*.
    make_as_duplicate (*another*: *ABSTRACT_DESCRIPTOR*)
       -- On creation, create a duplicate from another descriptor.
       -- As normal call, closes its own descriptor first (if open) and
       -- duplicates next.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
   -- Read and write to memory block
   last_blocked: *BOOLEAN*
      -- Would last call to *read* or *write* block?
   read (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
      -- Read data into *buf* at *offset* for *nbytes* bytes.
      -- The number of bytes actually read, is available in *last_read*.
   write (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
      -- Write given data from *buf* at *offset*, for *nbytes*
      -- bytes. Number of actually written bytes are in
      -- *last_written*. *last_written* can be unequal to *nbytes*
      -- if i/o is non-blocking or some error has occurred.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
   -- Eiffel like output
   put (*a*: *ANY*)
      -- Write any Eiffel object as string using its *out* value.
   write_character (*c*: *CHARACTER*)
      -- Write a character.
   write_string (*a_string*: *STRING*)
      -- Write *a_string* to output stream.
   puts (*a_string*: *STRING*)
      -- Write *a_string* to output stream.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
   -- Buffered input
   read_character
      -- Sets *last_character*.
      -- If this routine blocks, *last_character* has the value
      -- %U. Therefore, if non-blocking is enabled, always check
      -- *last_blocked* to see if the value make sense.
   read_string (*nb*: *INTEGER*)
      -- Read at most *nb* characters from input stream.
      -- Make the characters that have actually been read
      -- available in *last_string*.
      -- Zero characters will be read when non-blocking i/o
      -- is enabled, and *read* would block.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
   -- Status report
   is_attached_to_terminal: *BOOLEAN*
      -- Is the handle associated with character device?
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
   -- Access

value: *INTEGER*
-- The actual file descriptor value
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
-- non-blocking i/o
is_blocking_io: *BOOLEAN*
   require
     open: *is_open_read*
set_blocking_io (*enable*: *BOOLEAN*)
   require
     supports_nonblocking_io: **not** *enable* **implies** *supports_nonblocking_io*;
     open: *is_open*
   ensure
     blocking_set: *enable* = *is_blocking_io*
supports_nonblocking_io: *BOOLEAN*
feature(*s*) **from** *ABSTRACT_SOCKET*
-- Status
supports_receive_buffer_size: *BOOLEAN*
   -- Does this socket implementation support querying and
   -- setting the receive buffer size?
   -- Supported on all platforms except BeOS
supports_send_buffer_size: *BOOLEAN*
   -- Does this socket implementation support querying and
   -- setting the send buffer size?
   -- Supported on all platforms except BeOS
feature(*s*) **from** *ABSTRACT_SOCKET*
-- Access
receive_buffer_size: *INTEGER*
   -- Size of receive buffer;
   -- Not supported on BeOS.
send_buffer_size: *INTEGER*
   -- Size of send buffer
   -- Not supported on BeOS.
feature(*s*) **from** *ABSTRACT_SOCKET*
-- Change
set_receive_buffer_size (*a_new_size*: *INTEGER*)
   -- Set size of receive buffer to at least *a_new_size*.
set_send_buffer_size (*a_new_size*: *INTEGER*)
   -- Set size of send buffer to at least *a_new_size*.
feature(*s*) **from** *ABSTRACT_SOCKET*
-- Callbacks for the Multiplexer
multiplexer_read_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
   -- callback for read
multiplexer_write_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
   -- callback for read
multiplexer_error_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
   -- callback for read
multiplexer_read_idle_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
   -- callback for read
multiplexer_write_idle_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)

-- callback for read

feature(*s*) **from** *ABSTRACT_INTERNET_SOCKET*

-- Change

set_low_delay

-- Minimize delay.

set_throughput

-- Maximize throughput.

feature(*s*) **from** *ABSTRACT_INTERNET_SOCKET*

-- Local and remote addresses

local_address: *ABSTRACT_SOCKET_ADDRESS_IN_BASE*

-- Return address used on this side to talk to remote.

remote_address: *ABSTRACT_SOCKET_ADDRESS_IN_BASE*

-- Return address used at remote side to talk to this side.

feature(*s*) **from** *ABSTRACT_TCP_SOCKET*

-- Shutdown

shutdown_read

-- The read-half of the connection is closed. No more data

-- can be received on the socket and any data currently in

-- the socket receive buffer is discarded. The process can no

-- longer issue any of the read functions on the socket. Any

-- data received after this call for a TCP socket is

-- acknowledged and then silently discarded.

shutdown_read_write

-- The read-half and write-half of the connection are both

-- closed. This is equivalent to calling *shutdown-read* and

-- *shutdown_write*.

shutdown_write

-- The write-half of the connection is closed. In the case of

-- TCP, this is called a half-close. Any data currently in

-- the socket send buffer will be sent, followed by TCPs

-- normal connection termination sequence. The process can no

-- longer issue any of the write functions on the socket.

feature(*s*) **from** *ABSTRACT_TCP_SOCKET*

-- Socket options

set_nodelay

-- Disable TCPs Nagle algorithm. By default this algorithm

-- is enabled.

feature(*s*) **from** *ABSTRACT_TCP_CLIENT_SOCKET*

-- Socket specific open functions

open_by_address (*hp*: *EPX_HOST_PORT*)

-- Open socket to server specified in *hp*.

open_by_name_and_port (*a_host_name*: *STRING*; *a_port*: *INTEGER*)

-- Initialize given a server name and port.

-- If *a_host_name* is an ip address, the result is unspecified.

-- If *a_host_name* cannot be resolved, an exception is thrown.

invariant

accessing_real_singleton: *security_is_real_singleton*;

valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;

open_in_sync: *is_open_read* **or** *is_open_write* **implies** *is_open*; -- The reverse is not true, for examples sockets can be
 -- closed for reading/writing, but still open.
capacity_not_negative: *capacity* >= *0*;
valid_capacity: *is_open* = (*capacity* > *0*);
open_implies_handle_assigned: *is_open* = (*fd* /= *unassigned_value*);
owned_implies_open: *is_owner* **implies** *is_open*;
owned_implies_handle_assigned: *is_owner* **implies** *fd* /= *unassigned_value*;
line_buffer_index_offset_ok: *line_buffer* /= *Void* **implies** *line_buffer_index* <= *line_buffer.count*;
end *of* **deferred** *ABSTRACT_TCP_CLIENT_SOCKET*

B.12   Short form of ABSTRACT_TCP_SERVER_SOCKET

deferred class *interface ABSTRACT_TCP_SERVER_SOCKET*
feature(*s*) **from** *MEMORY*
   dispose
     -- Close handle if owner.
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Input
   non_blocking_read_character
     -- Read the next item in input stream.
     -- Make the result available in *last_item*.
   non_blocking_read_to_buffer (*a_buffer*: *KI_BUFFER*[*CHARACTER*]; *pos, nb*: *INTEGER*):
INTEGER
     -- Fill *a_buffer*, starting at position *pos*, with
     -- at most *nb* items read from input stream.
     -- Return the number of items actually read.
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Status report
   is_closable_for_reading: *BOOLEAN*
     -- Can current input stream be closed?
   is_open_read: *BOOLEAN*
     -- Can items be read from input stream?
   is_rewindable: *BOOLEAN*
     -- Can current input stream be rewound to return input from
     -- the beginning of the stream?
   end_of_input: *BOOLEAN*
     -- Has end-of-file been reached?
   valid_unread_character (*a_character*: *CHARACTER*): *BOOLEAN*
     -- Can *a_character* be put back in input stream?
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Access
   name: *STDC_PATH*
     -- Scratch path
   last_character: *CHARACTER*
     -- Last character read by *read_character* and a few other
     -- routines
feature(*s*) **from** *KI_INPUT_STREAM*
   -- Basic operations
   close_for_reading
     -- Try to close input stream if it is closable. Set
     -- *is_open_read* to false if operation was successful.
   rewind
     -- Move input position to the beginning of stream.
feature(*s*) **from** *KL_IMPORTED_ANY_ROUTINES*
   -- Access
   *any_*: *KL_ANY_ROUTINES*
     -- Routines that ought to be in class ANY
feature(*s*) **from** *KI_CHARACTER_INPUT_STREAM*
   -- Input

non__blocking__read__string (*nb*: *INTEGER*)
     -- Read at most *nb* characters from input stream.
     -- Make the characters that have actually been read
     -- available in *last__string*.
non__blocking__read__to__string (*a__string*: *STRING*; *pos, nb*: *INTEGER*): *INTEGER*
     -- Fill *a__string*, starting at position *pos*, with
     -- at most *nb* characters read from input stream.
     -- Return the number of characters actually read.
feature(*s*) **from** *KI__CHARACTER__INPUT__STREAM*
   -- Access
last__string: *STRING*
     -- Last string read;
     -- (Note: this query always return the same object.
     -- Therefore a clone should be used if the result
     -- is to be kept beyond the next call to this feature.
     -- However *last__string* is not shared between file objects.)
feature(*s*) **from** *EPX__CHARACTER__INPUT__STREAM*
   -- Access
is__streaming: *BOOLEAN*
     -- Is data coming from a network stream?
feature(*s*) **from** *EPX__CHARACTER__INPUT__STREAM*
   -- Input
last__read: *INTEGER*
     -- Last bytes read by *read__buffer*;
     -- Can be less than requested for non-blocking input.
     -- Check *last__blocked* in that case.
read__buffer (*buf*: *STDC__BUFFER*; *offset, nbytes*: *INTEGER*)
     -- Read data into *buf* at *offset* for *nbytes* bytes.
     -- Number of bytes actually read are available in *last__read*.
     -- This is a more safe version of *read* in case you have a
     -- STDC__BUFFER object.
feature(*s*) **from** *KI__TEXT__INPUT__STREAM*
   -- Input
read__line
     -- Read characters from input stream until a line separator
     -- or end of file is reached. Make the characters that have
     -- been read available in *last__string* and discard the line
     -- separator characters from the input stream.
     -- Zero characters will be read when non-blocking i/o
     -- is enabled, and *read__line* would block at the first character.
     -- If a character has been read, *read__line* will block until
     -- a %N has been read or *end__of__input* occurs.
read__new__line
     -- Read a line separator from input file.
     -- Make the characters making up the recognized
     -- line separator available in *last__string*,
     -- or make *last__string* empty and leave the
     -- input file unchanged if no line separator
     -- was found.

feature(*s*) **from** *KI_TEXT_INPUT_STREAM*
   -- Access
   eol: *STRING*
     -- Line separator;
     -- EPX classes do not distinguish between a %R%N or just %N
     -- end-of-line. The platform may though.
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
   -- The singleton, available to any because its used in preconditions
   security: *STDC_SECURITY*
     -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
   -- Access
   errno: *STDC_ERRNO*
     -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
   -- Status
   raise_exception_on_error: *BOOLEAN*
     -- Should an exception be raised when an error occurs?
     -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
   -- Change
   set_default_action_on_error
     -- Use *security.error_handling.exceptions_enabled* to
     -- determine if an exception should be raised when a C call
     -- returns an error.
   set_raise_exception_on_error
     -- Always raise an exception when a C call returns an error.
   set_continue_on_error
     -- Never raise an exception when a C call returns an error.
   inherit_error_handling (*an_instance*: *STDC_BASE*)
     -- Handle errors like *an_instance*
feature(*s*) **from** *KI_OUTPUT_STREAM*
   -- Output
   put_character (*c*: *CHARACTER*)
     -- Write a character.
   append (*an_input_stream*: *KI_INPUT_STREAM*[*CHARACTER*])
     -- Read items of *an_input_stream* until the end
     -- of input is reached, and write these items to
     -- current output stream.
     -- *append* is safe for non-blocking descriptors.
feature(*s*) **from** *KI_OUTPUT_STREAM*
   -- Status report
   is_open_write: *BOOLEAN*
     -- Can items be written to output stream?
   is_closable_for_writing: *BOOLEAN*
     -- Can current output stream be closed?
feature(*s*) **from** *KI_OUTPUT_STREAM*
   -- Basic operations
   close_for_writing

-- Try to close output stream if it is closable. Set
-- *is_open_write* to false if operation was successful.
feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
-- Output
put_string (*a_string*: *STRING*)
-- Write *a_string* to output stream.
put_substring (*a_string*: *STRING*; *s, e*: *INTEGER*)
-- Write substring of *a_string* between indexes
-- *s* and *e* to output stream.
put_integer (*i*: *INTEGER*)
-- Write decimal representation
-- of *i* to output stream.
-- Regexp: 0|(-?[1-9][0-9]*)
put_boolean (*b*: *BOOLEAN*)
-- Write "True" to output stream if
-- *b* is true, "False" otherwise.
feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
-- Basic operations
flush
-- Flush buffered data to disk.
feature(*s*) **from** *EPX_CHARACTER_OUTPUT_STREAM*
-- Output
last_written: *INTEGER*
-- How many bytes were written by the last call to a routine;
-- Can be less than requested for non-blocking output.
-- Check *last_blocked* in that case.
put_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
-- More safe version of *write* in case you have a
-- STDC_BUFFER object.
write_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
-- More safe version of *write* in case you have a
-- STDC_BUFFER object.
feature(*s*) **from** *KI_TEXT_OUTPUT_STREAM*
-- Output
put_line (*a_string*: *STRING*)
-- Write *a_string* to output stream
-- followed by a line separator.
put_new_line
-- Write a line separator to output stream.
feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*
-- Basic operations
close
-- Close the resource.
feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*
-- Status report
is_closable: *BOOLEAN*
-- Can current stream be closed for reading and writing?
is_open: *BOOLEAN*
-- Does *handle* contain an open handle?

is_owner: *BOOLEAN*
    -- Does this object close the stream on *close* or *dispose*?
    -- Only for resources that are owned, are resource limits checked.
feature(*s*) **from** *STDC_HANDLE*
  -- Access
resource_usage_can_be_increased: *BOOLEAN*
    -- Is it allowed to open another file?
feature(*s*) **from** *STDC_HANDLE*
  -- Influence ownership of the handle. Can help to influence subtle garbage collector problems
become_owner
    -- This class will own its handle. This is the only function
    -- that actually increases the resource count.
unown
    -- Resource will not be closed on dispose. Calling close will
    -- be forbidden. This routine may not call any other object,
    -- else it cannot be called from within dispose.
feature(*s*) **from** *STDC_HANDLE*
  -- Close
detach
    -- Forget the resource. Resource is not closed.
    -- You cannot read and write anymore.
feature(*s*) **from** *STDC_HANDLE*
  -- Resource
capacity: *INTEGER*
    -- Number of resources that are in use by *handle*. For a
    -- file this is 1, for a memory handle, this is the number of
    -- bytes.
fd: *H*
    -- Identifier of resource tracked by this class.
feature(*s*) **from** *PORTABLE_PATH*
  -- Change
set_portable_path (*a_path*: *STRING*)
    -- Set *portable_path* to *a_path*.
feature(*s*) **from** *HASHABLE*
hash_code: *INTEGER*
    -- Hash code value
feature(*s*) **from** *STDC_HANDLE_BASED_IO_STREAM*
  -- Stream or disk file
set_streaming (*enable*: *BOOLEAN*)
    -- Influence behaviour of certain functions if they should be
    -- optimized for data coming from disk or data coming from
    -- the network. In particular *is_streaming* implies that a
    -- client application is prepared to handle *read*s that
    -- return less than the requested number of bytes, but dont
    -- assume that means end-of-file.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Initialization
make
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*

-- Special creation
attach__to__socket (*a__fd*: *INTEGER*; *a__become__owner*: *BOOLEAN*)
    -- Create descriptor with value *a__fd*. Descriptor will close
    -- it when *a__become__owner*.
make__as__duplicate (*another*: *ABSTRACT__DESCRIPTOR*)
    -- On creation, create a duplicate from another descriptor.
    -- As normal call, closes its own descriptor first (if open) and
    -- duplicates next.
feature(*s*) **from** *ABSTRACT__DESCRIPTOR*
-- Read and write to memory block
last__blocked: *BOOLEAN*
    -- Would last call to *read* or *write* block?
read (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
    -- Read data into *buf* at *offset* for *nbytes* bytes.
    -- The number of bytes actually read, is available in *last__read*.
write (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
    -- Write given data from *buf* at *offset*, for *nbytes*
    -- bytes. Number of actually written bytes are in
    -- *last__written*. *last__written* can be unequal to *nbytes*
    -- if i/o is non-blocking or some error has occurred.
feature(*s*) **from** *ABSTRACT__DESCRIPTOR*
-- Eiffel like output
put (*a*: *ANY*)
    -- Write any Eiffel object as string using its *out* value.
write__character (*c*: *CHARACTER*)
    -- Write a character.
write__string (*a__string*: *STRING*)
    -- Write *a__string* to output stream.
puts (*a__string*: *STRING*)
    -- Write *a__string* to output stream.
feature(*s*) **from** *ABSTRACT__DESCRIPTOR*
-- Buffered input
read__character
    -- Sets *last__character*.
    -- If this routine blocks, *last__character* has the value
    -- %U. Therefore, if non-blocking is enabled, always check
    -- *last__blocked* to see if the value make sense.
read__string (*nb*: *INTEGER*)
    -- Read at most *nb* characters from input stream.
    -- Make the characters that have actually been read
    -- available in *last__string*.
    -- Zero characters will be read when non-blocking i/o
    -- is enabled, and *read* would block.
feature(*s*) **from** *ABSTRACT__DESCRIPTOR*
-- Status report
is__attached__to__terminal: *BOOLEAN*
    -- Is the handle associated with character device?
feature(*s*) **from** *ABSTRACT__DESCRIPTOR*
-- Access

value: *INTEGER*
     -- The actual file descriptor value
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
   -- non-blocking i/o
   is_blocking_io: *BOOLEAN*
      require
         open: *is_open_read*
   set_blocking_io (*enable*: *BOOLEAN*)
      require
         supports_nonblocking_io: **not** *enable* **implies** *supports_nonblocking_io*;
         open: *is_open*
      ensure
         blocking_set: *enable* = *is_blocking_io*
   supports_nonblocking_io: *BOOLEAN*
feature(*s*) **from** *ABSTRACT_SOCKET*
   -- Status
   supports_receive_buffer_size: *BOOLEAN*
      -- Does this socket implementation support querying and
      -- setting the receive buffer size?
      -- Supported on all platforms except BeOS
   supports_send_buffer_size: *BOOLEAN*
      -- Does this socket implementation support querying and
      -- setting the send buffer size?
      -- Supported on all platforms except BeOS
feature(*s*) **from** *ABSTRACT_SOCKET*
   -- Access
   receive_buffer_size: *INTEGER*
      -- Size of receive buffer;
      -- Not supported on BeOS.
   send_buffer_size: *INTEGER*
      -- Size of send buffer
      -- Not supported on BeOS.
feature(*s*) **from** *ABSTRACT_SOCKET*
   -- Change
   set_receive_buffer_size (*a_new_size*: *INTEGER*)
      -- Set size of receive buffer to at least *a_new_size*.
   set_send_buffer_size (*a_new_size*: *INTEGER*)
      -- Set size of send buffer to at least *a_new_size*.
feature(*s*) **from** *ABSTRACT_SOCKET*
   -- Callbacks for the Multiplexer
   multiplexer_read_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
      -- callback for read
   multiplexer_write_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
      -- callback for read
   multiplexer_error_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
      -- callback for read
   multiplexer_read_idle_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)
      -- callback for read
   multiplexer_write_idle_callback (*a_multiplexer*: *EPX_SOCKET_MULTIPLEXER*)

-- callback for read

feature(*s*) **from** *ABSTRACT_INTERNET_SOCKET*

-- Change

set_low_delay

-- Minimize delay.

set_throughput

-- Maximize throughput.

feature(*s*) **from** *ABSTRACT_INTERNET_SOCKET*

-- Local and remote addresses

local_address: *ABSTRACT_SOCKET_ADDRESS_IN_BASE*

-- Return address used on this side to talk to remote.

remote_address: *ABSTRACT_SOCKET_ADDRESS_IN_BASE*

-- Return address used at remote side to talk to this side.

feature(*s*) **from** *ABSTRACT_TCP_SOCKET*

-- Shutdown

shutdown_read

-- The read-half of the connection is closed. No more data

-- can be received on the socket and any data currently in

-- the socket receive buffer is discarded. The process can no

-- longer issue any of the read functions on the socket. Any

-- data received after this call for a TCP socket is

-- acknowledged and then silently discarded.

shutdown_read_write

-- The read-half and write-half of the connection are both

-- closed. This is equivalent to calling *shutdown-read* and

-- *shutdown_write*.

shutdown_write

-- The write-half of the connection is closed. In the case of

-- TCP, this is called a half-close. Any data currently in

-- the socket send buffer will be sent, followed by TCPs

-- normal connection termination sequence. The process can no

-- longer issue any of the write functions on the socket.

feature(*s*) **from** *ABSTRACT_TCP_SOCKET*

-- Socket options

set_nodelay

-- Disable TCPs Nagle algorithm. By default this algorithm

-- is enabled.

feature(*s*) **from** *ABSTRACT_TCP_SERVER_SOCKET*

-- Accept

accept: *ABSTRACT_TCP_SOCKET*

-- Return the next completed connection from the front of the

-- completed connection queue. If there are no completed

-- connections, the process is put to sleep.

-- If the socket is non-blocking, Void will be returned and

-- the process is not put to sleep..

last_client_address: *ABSTRACT_SOCKET_ADDRESS_IN_BASE*

-- Address of last client accepted by *accept*.

invariant

accessing_real_singleton: *security_is_real_singleton*;

_valid_error_action_: _error_action_ $>=$ _0_ **and** _error_action_ $<=$ _2_;
_open_in_sync_: _is_open_read_ **or** _is_open_write_ **implies** _is_open_; -- The reverse is not true, for examples sockets can be
 -- closed for reading/writing, but still open.
_capacity_not_negative_: _capacity_ $>=$ _0_;
_valid_capacity_: _is_open_ $=$ (_capacity_ $>$ _0_);
_open_implies_handle_assigned_: _is_open_ $=$ (_fd_ $/=$ _unassigned_value_);
_owned_implies_open_: _is_owner_ **implies** _is_open_;
_owned_implies_handle_assigned_: _is_owner_ **implies** _fd_ $/=$ _unassigned_value_;
_line_buffer_index_offset_ok_: _line_buffer_ $/=$ _Void_ **implies** _line_buffer_index_ $<=$ _line_buffer.count_;
_client_socket_address_not_void_: _client_socket_address_ $/=$ _Void_;
end _of_ **deferred** _ABSTRACT_TCP_SERVER_SOCKET_

C
Short (flat)
listing of
POSIX classes

C.1 Short form of POSIX_ASYNC_IO_REQUEST

class *interface POSIX_ASYNC_IO_REQUEST*
creation
  make (a_fd: POSIX_FILE_DESCRIPTOR)
feature(*s*) **from** *POSIX_ASYNC_IO_REQUEST*
  -- creation
  make (a_fd: POSIX_FILE_DESCRIPTOR)
feature(*s*) **from** *POSIX_ASYNC_IO_REQUEST*
  -- request properties
  raw_pointer: POINTER
    -- Location for read or written data, usually *buffer* is a
    -- better idea.
  count: INTEGER
    -- number of bytes to read/write
  offset: INTEGER
    -- file offset
feature(*s*) **from** *POSIX_ASYNC_IO_REQUEST*

-- set request properties
set_buffer (*a_buffer*: *STDC_BUFFER*)
   -- set memory location to read/write from.
set_count (*a_count*: *INTEGER*)
   -- set number of bytes to read/write
set_offset (*a_offset*: *INTEGER*)
set_raw_pointer (*a_pointer*: *POINTER*)
   -- set memory location to read/write from. Make sure you have
   -- called *set_count* first!
feature(*s*) **from** *POSIX_ASYNC_IO_REQUEST*
   -- basic read/write requests
read
   -- execute async read request
write
   -- execute async write request
feature(*s*) **from** *POSIX_ASYNC_IO_REQUEST*
   -- Eiffel friendly reads and writes
last_string: *STRING*
   -- attempt to return buffer as an Eiffel string
   -- buffer should have a terminating byte!
read_string
put_string (*text*: *STRING*)
write_string (*text*: *STRING*)
feature(*s*) **from** *POSIX_ASYNC_IO_REQUEST*
   -- other operations
cancel_failed: *BOOLEAN*
   -- set by cancel, True if cancel request failed, probably
   -- because operation was already performed
cancel
   -- cancel request
synchronize
   -- force all i/o operations queued for the file descriptor
   -- associated with this request to the synchronous state.
   -- Function returns when the request has been initiated or
   -- queued to the file or device (even when the data cannot be
   -- synchronized immediately)
synchronize_data
   -- force all i/o operations queued for the file descriptor
   -- associated with this request to the synchronous state.
   -- Function returns when the request has been initiated or
   -- queued to the file or device (even when the data cannot be
   -- synchronized immediately)
wait_for
   -- suspend process, until request completed
feature(*s*) **from** *POSIX_ASYNC_IO_REQUEST*
   -- Access
buffer: *STDC_BUFFER*
   -- Buffer where data that is being read/write comes from,
   -- unless set_pointer has been called

fd: *POSIX__FILE__DESCRIPTOR*
is_pending: *BOOLEAN*
    -- Is io request still pending?
return_status: *INTEGER*
    -- Return status of asynchronous i/o operation, equal to what
    -- the synchronous read, write of fsync would have returned
invariant
accessing_real_singleton: *security_is_real_singleton*;
valid_error_action: *error_action* $>=$ *0* **and** *error_action* $<=$ *2*;
valid_aiocb: *aiocb* $/=$ *Void*;
synced_buffer_and_raw_pointer: *buffer* $/=$ *Void* **implies** *buffer.ptr = raw_pointer*;
end *of POSIX__ASYNC__IO__REQUEST*

C.2   Short form of POSIX_BASE

class *interface POSIX_BASE*
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action >= 0* **and** *error_action <= 2*;
end *of POSIX_BASE*

C.3 Short form of POSIX__CHILD__PROCESS

deferred class *interface POSIX__CHILD__PROCESS*
feature(*s*) **from** *POSIX__CHILD__PROCESS*
   -- Childs pid
   pid: *INTEGER*
     -- The process identifier.
   is__pid__valid: *BOOLEAN*
     -- return True if this object refers to a child process, so
     -- it has an id
feature(*s*) **from** *POSIX__CHILD__PROCESS*
   -- Actions that parent may execute
   wait__for (*suspend*: *BOOLEAN*)
     -- Wait for this process to terminate. If *suspend* then we
     -- wait until the information about this process is available,
     -- else we return immediately.
     -- If suspend is False, check the running property to see
     -- if this child is really terminated.
invariant
   --2007-12-13: invariant failure in some cases, root cause not determined yet
   --pid__known__is__not__terminated: is__pid__valid = not is__terminated
   accessing__real__singleton: *security__is__real__singleton*;
   valid__error__action: *error__action* >= *0* **and** *error__action* <= *2*;
end *of* **deferred** *POSIX__CHILD__PROCESS*

C.4 Short form of POSIX_CONSTANTS

class *interface POSIX_CONSTANTS*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Error codes
   edom: *INTEGER*
     -- Math argument out of domain of function
   erange: *INTEGER*
     -- Math result not representable
   emfile: *INTEGER*
     -- Too many open files
feature(*s*) **from** *STDC_CONSTANTS*
   -- Standard streams
   stream_stdin: *POINTER*
   stream_stdout: *POINTER*
   stream_stderr: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Special characters
   const_eof: *INTEGER*
     -- signals EOF
feature(*s*) **from** *STDC_CONSTANTS*
   -- I/O buffering
   iofbf: *INTEGER*
     -- full buffering
   iolbf: *INTEGER*
     -- line buffering
   ionbf: *INTEGER*
     -- no buffering
feature(*s*) **from** *STDC_CONSTANTS*
   -- file positioning
   seek_set: *INTEGER*
   seek_cur: *INTEGER*
   seek_end: *INTEGER*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Signal related constants
   sig_dfl: *POINTER*
   sig_err: *POINTER*
   sig_ign: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
   -- Signals
   sigabrt: *INTEGER*
   sigfpe: *INTEGER*
     -- erroneous arithmetic operation, such as zero divide or an
     -- operation resulting in overflow
   sigill: *INTEGER*
     -- illegal instruction
   sigint: *INTEGER*
     -- receipt of an interactive attention signal
   sigsegv: *INTEGER*

        -- invalid access to storage
    sigterm: *INTEGER*
        -- Request process to terminate; can be caught or ignored
  feature(*s*) **from** *STDC_CONSTANTS*
    -- random numbers
    rand_max: *INTEGER*
        -- maximum value returned by the *random* function
  feature(*s*) **from** *STDC_CONSTANTS*
    -- category constants
    lc_ctype: *INTEGER*
    lc_numeric: *INTEGER*
    lc_time: *INTEGER*
    lc_collate: *INTEGER*
    lc_monetary: *INTEGER*
    lc_all: *INTEGER*
  feature(*s*) **from** *STDC_CONSTANTS*
    -- various
    clocks_per_sec: *INTEGER*
  feature(*s*) **from** *STDC_CONSTANTS*
    -- exit codes
    exit_failure: *INTEGER*
        -- exit status when something has gone wrong
    exit_success: *INTEGER*
        -- exit status upon success
  feature(*s*) **from** *POSIX_CONSTANTS*
    -- Error codes
    e2big: *INTEGER*
        -- Arg list too long
    eacces: *INTEGER*
        -- Permission denied
    eagain: *INTEGER*
        -- Resource temporarily unavailable
    ewouldblock: *INTEGER*
        -- Resource temporarily unavailable
    ebadf: *INTEGER*
        -- Bad file descriptor
    ebusy: *INTEGER*
        -- Resource busy
    ecanceled: *INTEGER*
        -- Operation canceled
    echild: *INTEGER*
        -- No child processes
    edeadlk: *INTEGER*
        -- Resource deadlock avoided
    eexist: *INTEGER*
        -- File exists
    efault: *INTEGER*
        -- Bad address
    efbig: *INTEGER*

-- File too large
einprogress: *INTEGER*
  -- Operation in progress
eintr: *INTEGER*
  -- Interrupted function call
einval: *INTEGER*
  -- Invalid argument
eio: *INTEGER*
  -- Input/output error
eisdir: *INTEGER*
  -- Is a directory
emlink: *INTEGER*
  -- Too many links
emsgsize: *INTEGER*
  -- Inappropriate message buffer length
enametoolong: *INTEGER*
  -- Filename too long
enfile: *INTEGER*
  -- Too many open files in system
enodev: *INTEGER*
  -- No such device
enoent: *INTEGER*
  -- No such file or directory
enoexec: *INTEGER*
  -- Exec format error
enolck: *INTEGER*
  -- No locks available
enomem: *INTEGER*
  -- Not enough space
enospc: *INTEGER*
  -- There is no free space remaining on the device
enosys: *INTEGER*
  -- Function not implemented
enotdir: *INTEGER*
  -- Not a directory
enotempty: *INTEGER*
  -- Directory not empty
enotsup: *INTEGER*
  -- Not supported
enotty: *INTEGER*
  -- Inappropriate I/O control operation
enxio: *INTEGER*
  -- No such device or address
eperm: *INTEGER*
  -- Operation not permitted
epipe: *INTEGER*
  -- Broken pipe
erofs: *INTEGER*
  -- Read-only file system

    espipe: *INTEGER*
       -- Invalid seek;
       -- An lseek() function was issued on a pipe or FIFO.
    esrch: *INTEGER*
       -- No such process
    etimedout: *INTEGER*
       -- Operation timed out
    exdev: *INTEGER*
       -- Improper link;
       -- A link to a file on another file system was attempted.
feature(*s*) **from** *POSIX_CONSTANTS*
    -- standard file numbers
    stderr_fileno: *INTEGER*
    stdin_fileno: *INTEGER*
    stdout_fileno: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
    -- posix open symbolic constants
    o_append: *INTEGER*
       -- Set the file offset to the end-of-file prior to each write
    o_creat: *INTEGER*
       -- If the file does not exist, allow it to be created. This
       -- flag indicates that the mode argument is present in the
       -- call to open.
    o_dsync: *INTEGER*
       -- Write according to synchronized i/o data integrity completion
    o_excl: *INTEGER*
       -- Open fails if the file already exists
    o_exclusive: *INTEGER*
       -- Open fails if the file already exists
    o_noctty: *INTEGER*
       -- prevents terminal from becoming the controlling terminal
       -- for this process
    o_nonblock: *INTEGER*
       -- Do not wait for device or file to be ready or available
    o_rdonly: *INTEGER*
       -- Open for reading only
    o_rdwr: *INTEGER*
       -- Open fo reading and writing
    o_rsync: *INTEGER*
       -- Synchronized read i/o operations
    o_sync: *INTEGER*
       -- Write according to synchronized i/o file integrity completion
    o_trunc: *INTEGER*
       -- Use only on ordinary files opened for writing. It causes
       -- the file to be truncated to zero length.
    o_wronly: *INTEGER*
       -- Open for writing only
feature(*s*) **from** *POSIX_CONSTANTS*
    -- posix permission symbolic constants

s_irusr: *INTEGER*
s_iread: *INTEGER*
s_iwusr: *INTEGER*
s_iwrite: *INTEGER*
s_ixusr: *INTEGER*
s_iexec: *INTEGER*
s_irgrp: *INTEGER*
s_iwgrp: *INTEGER*
s_ixgrp: *INTEGER*
s_iroth: *INTEGER*
s_iwoth: *INTEGER*
s_ixoth: *INTEGER*
s_isuid: *INTEGER*
s_isgid: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- Posix accessibility constants
f_ok: *INTEGER*
r_ok: *INTEGER*
w_ok: *INTEGER*
x_ok: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- Posix signal constants
sa_nocldstop: *INTEGER*
sighup: *INTEGER*
   -- hangup detected on controlling terminal or death of
   -- controlling process
signal_hangup: *INTEGER*
   -- hangup detected on controlling terminal or death of
   -- controlling process
sigalrm: *INTEGER*
   -- Timeout signal, such as initiated by the alarm() function
   -- or see POSIX_TIMED_COMMAND
signal_alarm: *INTEGER*
   -- Timeout signal, such as initiated by the alarm() function
   -- or see POSIX_TIMED_COMMAND
sigchld: *INTEGER*
   -- Child process terminated or stopped
signal_child: *INTEGER*
   -- Child process terminated or stopped
sigkill: *INTEGER*
   -- Termination signal (cannot be caught or ignored)
signal_kill: *INTEGER*
   -- Termination signal (cannot be caught or ignored)
sigpipe: *INTEGER*
   -- Write on a pipe with no readers
signal_pipe: *INTEGER*
   -- Write on a pipe with no readers
sigquit: *INTEGER*
   -- Interactive termination signal

signal_quit: *INTEGER*
   -- Interactive termination signal
sigcont: *INTEGER*
   -- Continue if stopped
signal_continue: *INTEGER*
   -- Continue if stopped
sigstop: *INTEGER*
   -- Stop signal, cannot be caught or ignored
signal_stop: *INTEGER*
   -- Stop signal, cannot be caught or ignored
sigtstp: *INTEGER*
   -- Interactive stop signal
signal_interactive_stop: *INTEGER*
   -- Interactive stop signal
sigttin: *INTEGER*
   -- Read from control terminal attempted by a member of a
   -- background process group
signal_terminal_in: *INTEGER*
   -- Read from control terminal attempted by a member of a
   -- background process group
sigttou: *INTEGER*
   -- Write to control terminal attempted by a member of a
   -- background process group
signal_terminal_out: *INTEGER*
   -- Write to control terminal attempted by a member of a
   -- background process group

feature(*s*) **from** *POSIX_CONSTANTS*
   -- sigprocmask how values
sig_block: *INTEGER*
sig_unblock: *INTEGER*
sig_setmask: *INTEGER*

feature(*s*) **from** *POSIX_CONSTANTS*
   -- Posix pathconf constants
pc_name_max: *INTEGER*
   -- The maximum length of a filename for this directory

feature(*s*) **from** *POSIX_CONSTANTS*
   -- terminal i/o local mode flags
isig: *INTEGER*
icanon: *INTEGER*
echo: *INTEGER*
   -- If set, input characters are echoed back to the terminal
echoe: *INTEGER*
echok: *INTEGER*
echonl: *INTEGER*
noflsh: *INTEGER*
tostop: *INTEGER*
iexten: *INTEGER*

feature(*s*) **from** *POSIX_CONSTANTS*
   -- set terminal settings options

tcsanow: *INTEGER*
tcsadrain: *INTEGER*
tcsaflush: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- Semaphore constants
sem_value_max: *INTEGER*
-- Valid maximum initial value for a semaphore.
feature(*s*) **from** *POSIX_CONSTANTS*
-- terminal baud rates
b0: *INTEGER*
b50: *INTEGER*
b75: *INTEGER*
b110: *INTEGER*
b134: *INTEGER*
b150: *INTEGER*
b200: *INTEGER*
b300: *INTEGER*
b600: *INTEGER*
b1200: *INTEGER*
b1800: *INTEGER*
b2400: *INTEGER*
b4800: *INTEGER*
b9600: *INTEGER*
b19200: *INTEGER*
b38400: *INTEGER*
b57600: *INTEGER*
b115200: *INTEGER*
b230400: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- terminal i/o control mode constants
csize: *INTEGER*
cs5: *INTEGER*
cs6: *INTEGER*
cs7: *INTEGER*
cs8: *INTEGER*
cstopb: *INTEGER*
cread: *INTEGER*
parenb: *INTEGER*
parodd: *INTEGER*
hupcl: *INTEGER*
clocal: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- terminal i/o input control flags
ignbrk: *INTEGER*
brkint: *INTEGER*
ignpar: *INTEGER*
parmrk: *INTEGER*
inpck: *INTEGER*
istrip: *INTEGER*

    inlcr: *INTEGER*
    igncr: *INTEGER*
    icrnl: *INTEGER*
    ixon: *INTEGER*
    ixoff: *INTEGER*
feature(*s*) **from** *POSIX__CONSTANTS*
    -- category constants
    lc__messages: *INTEGER*
feature(*s*) **from** *POSIX__CONSTANTS*
    -- pathname variable values
    max__input: *INTEGER*
        -- Minimum number of bytes for which space will be available
        -- in a terminal input queue; therefore, the maximum number
        -- of bytes a portable application may required to be typed
        -- as input before eading them
    name__max: *INTEGER*
        -- Maximum number of bytes in a file name
    path__max: *INTEGER*
        -- Maximum number of bytes in a pathname
    pipe__buf: *INTEGER*
        -- Maximum number of bytes that can be written atomically
        -- when writing to a pipe.
feature(*s*) **from** *POSIX__CONSTANTS*
    -- invariant values
    ssize__max: *INTEGER*
        -- The maximum value that can be stored in an object of type ssize__t
feature(*s*) **from** *POSIX__CONSTANTS*
    -- Other limits
    stream__max: *INTEGER*
        -- The number of streams that one process can have open at
        -- one time. If defined, it has the same value as {FOPEN__MAX}.
end *of POSIX__CONSTANTS*

C.5 Short form of POSIX_CURRENT_PROCESS

class *interface POSIX_CURRENT_PROCESS*
feature(*s*) **from** *ARGUMENTS*
  command_name: *STRING*
feature(*s*) **from** *CAPI_TIME*
  -- Standard C binding
  current_time: *INTEGER*
    -- The current calendar time in seconds since the epoch
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Process standard input/output/error
  stdin: *POSIX_TEXT_FILE*
  stdout: *POSIX_TEXT_FILE*
  stderr: *POSIX_TEXT_FILE*
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Various
  clock: *INTEGER*
    -- Approximation of processor time used by the program, or -1
    -- if unknown
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Random numbers
  random: *INTEGER*
    -- Returns a pseudo-random integer between 0 and *RAND_MAX*.
  set_random_seed (*a_seed*: *INTEGER*)
    -- Sets *a_seed* as the seed for a new sequence of
    -- pseudo-random integers to be returned by *random*. These
    -- sequences are repeatable by calling *set_random_seed* with
    -- the same seed value. If no seed value is provided, the
    -- *random* function is automatically seeded with a value of
    -- 1.
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Global locale
  locale: *STRING*
    -- Current locale
  numeric_format: *STDC_LOCALE_NUMERIC*
    -- Various information for formatting numbers and monetary
    -- quantities
  set_locale (*category*: *INTEGER*; *new_locale*: *STRING*)
    -- Set given locale to *new_locale*. *new_locale* is either a
    -- well-known constant like "C" or "da_DK" or an opaque
    -- string that was returned by another call of *setlocale*.
  set_c_locale
    -- Set locale to the Standard C locale (the default).
  set_native_decimal_point
    -- Set the decimal point character using the LC_NUMERIC
    -- environment variable.
  set_native_locale
    -- Set entire locale to the natives setting which is
    -- determend by environment variables like LC_NUMERIC,

      -- LC_COLLATE, LC_CTYPE etc.

set_native_time
      -- Set time display to the natives setting using the LC_TIME
      -- environment variable.

feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
  -- Access
effective_user_name: *STRING*
      -- Name of the user currently associated with the current
      -- thread;
      -- Name will not be Void, but can be empty if no name found
      -- (you can screw up your /etc/passwd on Unix...)
full_command_name: *STRING*
      -- *command_name* with fully qualified path;
      -- An empty string is returned in case *command_name* is
      -- empty. As any program can setup the arguments passed to
      -- another program, an empty *command_name* is a possibility.
pid: *INTEGER*
      -- Process identifier, unique for this process

feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
  -- Status
is_pid_valid: *BOOLEAN*
      -- Is *pid* valid?

feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
  -- Every process also has standard file descriptors which might not be compatible with
stdin/stdout/stderr (Windows)
fd_stdin: *POSIX_FILE_DESCRIPTOR*
fd_stdout: *POSIX_FILE_DESCRIPTOR*
fd_stderr: *POSIX_FILE_DESCRIPTOR*

feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
  -- Sleeping
millisleep (*a_milliseconds*: *INTEGER*)
      -- Sleep for *a_milliseconds* milliseconds. Due to timer
      -- resolution issues, the minimum resolution might be in the
      -- order of 10ms or higher.
sleep (*seconds*: *INTEGER*)
      -- Delays process execution up to *seconds*. Can return early
      -- if interrupted. Check *unslect_seconds*
unslept_seconds: *INTEGER*
      -- The number of seconds still to sleep, before being
      -- interrupted; it is set by *sleep*. If it is zero, no
      -- interrupt occurred and process slept for the allotted
      -- time.

feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
  -- The singleton, available to any because its used in preconditions
security: *STDC_SECURITY*
      -- Singleton entry point for security.

feature(*s*) **from** *STDC_BASE*
  -- Access
errno: *STDC_ERRNO*

      -- Access to the variable that contains the error that occurred.

feature(*s*) **from** *STDC_BASE*

  -- Status

raise_exception_on_error: *BOOLEAN*

    -- Should an exception be raised when an error occurs?

    -- If not, you have to check errno for any errors.

feature(*s*) **from** *STDC_BASE*

  -- Change

set_default_action_on_error

    -- Use *security.error_handling.exceptions_enabled* to

    -- determine if an exception should be raised when a C call

    -- returns an error.

set_raise_exception_on_error

    -- Always raise an exception when a C call returns an error.

set_continue_on_error

    -- Never raise an exception when a C call returns an error.

inherit_error_handling (*an_instance*: *STDC_BASE*)

    -- Handle errors like *an_instance*

feature(*s*) **from** *ABSTRACT_PROCESS*

  -- Signal this process

terminate

    -- attempt to gracefully terminate this process

feature(*s*) **from** *PAPI_WAIT*

  -- C binding functions

posix_wait (*statloc*: *POINTER*): *INTEGER*

    -- Waits for process termination

posix_waitpid (*a_pid*: *INTEGER*; *statloc*: *POINTER*; *options*: *INTEGER*): *INTEGER*

    -- Waits for process termination

feature(*s*) **from** *PAPI_WAIT*

  -- C binding statloc evaluation

posix_wexitstatus (*a_value*: *INTEGER*): *INTEGER*

    -- Evaluates to the low-order eight bits of the *status*

    -- argument that the child passed to exit, or the value the

    -- child process returned from main.

posix_wifexited (*a_value*: *INTEGER*): *BOOLEAN*

    -- Evaluates to a non-zero value if status was returned for

    -- a child that terminated normally

posix_wifsignaled (*a_value*: *INTEGER*): *BOOLEAN*

    -- Evaluates to a non-zero value if status was returned for

    -- a child that terminated due to the receipt of a signal

    -- that was not caught

posix_wifstopped (*a_value*: *INTEGER*): *BOOLEAN*

posix_wstopsig (*a_value*: *INTEGER*): *BOOLEAN*

posix_wtermsig (*a_value*: *INTEGER*): *INTEGER*

feature(*s*) **from** *PAPI_WAIT*

  -- waitpid contants

wnohang: *INTEGER*

    -- do not suspend execution

wuntraced: *INTEGER*

      -- report status of childs that are stopped and whose status has not

      -- yet been reported since they stopped

feature(*s*) **from** *EPX_CURRENT_PROCESS*

   -- Access (doesnt make a lot of sense if youre not inheriting)

   raw_environment_variables: *ARRAY*[*STRING*]

      -- The raw list of name=value pairs of environment

      -- variables passed to this process;

      -- A new list is created every time this feature is accessed.

feature(*s*) **from** *POSIX_PROCESS*

   -- signal this process

   kill (*a_signal_code*: *INTEGER*)

      -- Send signal *signal_code* to the process

feature(*s*) **from** *POSIX_CURRENT_PROCESS*

   -- POSIX locale specifics

   set_native_messages

      -- Select native language as the language in which messages

      -- are displayed.

invariant

   accessing_real_singleton: *security_is_real_singleton*;

   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;

end *of POSIX_CURRENT_PROCESS*

C.6 Short form of POSIX__DAEMON

deferred class *interface POSIX__DAEMON*
feature(*s*) **from** *POSIX__DAEMON*
    -- Daemon specific actions
    detach
        -- detach from command-line, not very useful if you want to
        -- spawn multiple daemons, but you can always pass daemons to
        -- the fork routine yourself.
    after__fork
        -- Code thanks to W. Richard Stevens.
        -- If you are started from inetd, youre in big trouble
        -- already and sinking deeper in the mud. For inetd there will
        -- be another method to call, perhaps *init__inetd* or so.
invariant
    --2007-12-13: invariant failure in some cases, root cause not determined yet
    --pid__known__is__not__terminated: is__pid__valid = not is__terminated
    accessing__real__singleton: *security__is__real__singleton*;
    valid__error__action: *error__action >= 0* **and** *error__action <= 2*;
end *of* **deferred** *POSIX__DAEMON*

C.7 Short form of POSIX_DIRECTORY

class *interface POSIX_DIRECTORY*
creation
   make (a_directory_name: STRING)
     -- Initialize for browsing *a_directory_name.*
invariant
   accessing_real_singleton: security_is_real_singleton;
   valid_error_action: error_action >= 0 **and** *error_action <= 2;*
   directory_name_not_empty: directory_name /= Void **and then not** *directory_name.is_empty;*
   my_status_tracks_item: my_status /= Void **implies** *my_status.path.is_equal(full_name);*
end *of POSIX_DIRECTORY*

C.8 Short form of POSIX_EXEC_PROCESS

class *interface POSIX_EXEC_PROCESS*
creation
   make (*a_program*: *STRING*; *a_arguments*: *ARRAY*[*STRING*])
   make_capture_input (*a_program*: *STRING*; *a_arguments*: *ARRAY*[*STRING*])
   make_capture_output (*a_program*: *STRING*; *a_arguments*: *ARRAY*[*STRING*])
   make_capture_io (*a_program*: *STRING*; *a_arguments*: *ARRAY*[*STRING*])
     -- Why not use three directional i/o, because youre getting
     -- yourself in great, great trouble anyway.
     -- A bit of advice: call stdin.close before starting to call
     -- stdout.read_string. But: your pipe might not have a large
     -- enough buffer, so you write to the process stdin and get
     -- blocked, because the process must empty its stdin
     -- first. The process will do that, but next write to
     -- stdout. If the stdout buffer is full, the process will
     -- block. Now we have a nice dead-lock. Happy coding.
   make_capture_all (*a_program*: *STRING*; *a_arguments*: *ARRAY*[*STRING*])
     -- Three directional i/o is a great way to get yourself in trouble.
feature(*s*) **from** *STDC_CHILD_PROCESS*
  -- Termination info
  has_exit_code: *BOOLEAN*
    -- Does *exit_code* return a valid value?
  is_terminated: *BOOLEAN*
    -- Is child not running any more?
  exit_code: *INTEGER*
    -- Low-order 8 bits of call to _exit or exit for this process
feature(*s*) **from** *ABSTRACT_CHILD_PROCESS*
  -- Actions that parent may execute
  wait_for (*suspend*: *BOOLEAN*)
    -- Wait for this process to terminate. If *suspend* then we
    -- wait until the information about this process is available,
    -- else we return immediately.
    -- If *suspend* is False, check *is_terminated* to see
    -- if this child is really terminated.
feature(*s*) **from** *ARGUMENTS*
  command_name: *STRING*
feature(*s*) **from** *CAPI_TIME*
  -- Standard C binding
  current_time: *INTEGER*
    -- The current calendar time in seconds since the epoch
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Process standard input/output/error
  child_stdin: *POSIX_TEXT_FILE*
  child_stdout: *POSIX_TEXT_FILE*
  child_stderr: *POSIX_TEXT_FILE*
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Various
  clock: *INTEGER*

        -- Approximation of processor time used by the program, or -1
        -- if unknown
  feature(*s*) **from** *STDC__CURRENT__PROCESS*
    -- Random numbers
    random: *INTEGER*
        -- Returns a pseudo-random integer between 0 and *RAND__MAX*.
    set__random__seed (*a__seed*: *INTEGER*)
        -- Sets *a__seed* as the seed for a new sequence of
        -- pseudo-random integers to be returned by *random*. These
        -- sequences are repeatable by calling *set__random__seed* with
        -- the same seed value. If no seed value is provided, the
        -- *random* function is automatically seeded with a value of
        -- 1.
  feature(*s*) **from** *STDC__CURRENT__PROCESS*
    -- Global locale
    locale: *STRING*
        -- Current locale
    numeric__format: *STDC__LOCALE__NUMERIC*
        -- Various information for formatting numbers and monetary
        -- quantities
    set__locale (*category*: *INTEGER*; *new__locale*: *STRING*)
        -- Set given locale to *new__locale. new__locale* is either a
        -- well-known constant like "C" or "da__DK" or an opaque
        -- string that was returned by another call of *setlocale.*
    set__c__locale
        -- Set locale to the Standard C locale (the default).
    set__native__decimal__point
        -- Set the decimal point character using the LC__NUMERIC
        -- environment variable.
    set__native__locale
        -- Set entire locale to the natives setting which is
        -- determend by environment variables like LC__NUMERIC,
        -- LC__COLLATE, LC__CTYPE etc.
    set__native__time
        -- Set time display to the natives setting using the LC__TIME
        -- environment variable.
  feature(*s*) **from** *ABSTRACT__CURRENT__PROCESS*
    -- Access
    effective__user__name: *STRING*
        -- Name of the user currently associated with the current
        -- thread;
        -- Name will not be Void, but can be empty if no name found
        -- (you can screw up your /etc/passwd on Unix...)
    full__command__name: *STRING*
        -- *command__name* with fully qualified path;
        -- An empty string is returned in case *command__name* is
        -- empty. As any program can setup the arguments passed to
        -- another program, an empty *command__name* is a possibility.
    child__pid: *INTEGER*

-- The process identifier.

feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*

  -- Status

  is_child_pid_valid: *BOOLEAN*

    -- return True if this object refers to a child process, so

    -- it has an id

feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*

  -- Every process also has standard file descriptors which might not be compatible with stdin/stdout/stderr (Windows)

  child_fd_stin: *POSIX_FILE_DESCRIPTOR*

  child_fd_stdout: *POSIX_FILE_DESCRIPTOR*

  child_fd_sterr: *POSIX_FILE_DESCRIPTOR*

feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*

  -- Sleeping

  millisleep (*a_milliseconds*: *INTEGER*)

    -- Sleep for *a_milliseconds* milliseconds. Due to timer

    -- resolution issues, the minimum resolution might be in the

    -- order of 10ms or higher.

  sleep (*seconds*: *INTEGER*)

    -- Delays process execution up to *seconds*. Can return early

    -- if interrupted. Check *unslect_seconds*

  unslept_seconds: *INTEGER*

    -- The number of seconds still to sleep, before being

    -- interrupted; it is set by *sleep*. If it is zero, no

    -- interrupt occurred and process slept for the allotted

    -- time.

feature(*s*) **from** *STDC_SECURITY_ACCESSOR*

  -- The singleton, available to any because its used in preconditions

  security: *STDC_SECURITY*

    -- Singleton entry point for security.

feature(*s*) **from** *STDC_BASE*

  -- Access

  errno: *STDC_ERRNO*

    -- Access to the variable that contains the error that occurred.

feature(*s*) **from** *STDC_BASE*

  -- Status

  raise_exception_on_error: *BOOLEAN*

    -- Should an exception be raised when an error occurs?

    -- If not, you have to check errno for any errors.

feature(*s*) **from** *STDC_BASE*

  -- Change

  set_default_action_on_error

    -- Use *security.error_handling.exceptions_enabled* to

    -- determine if an exception should be raised when a C call

    -- returns an error.

  set_raise_exception_on_error

    -- Always raise an exception when a C call returns an error.

  set_continue_on_error

    -- Never raise an exception when a C call returns an error.

inherit_error_handling (*an_instance*: *STDC_BASE*)
    -- Handle errors like *an_instance*
feature(*s*) **from** *ABSTRACT_PROCESS*
  -- Signal this process
terminate
    -- attempt to gracefully terminate this process
feature(*s*) **from** *EPX_CURRENT_PROCESS*
  -- Access (doesnt make a lot of sense if youre not inheriting)
raw_environment_variables: *ARRAY*[*STRING*]
    -- The raw list of name=value pairs of environment
    -- variables passed to this process;
    -- A new list is created every time this feature is accessed.
feature(*s*) **from** *POSIX_PROCESS*
  -- signal this process
kill (*a_signal_code*: *INTEGER*)
    -- Send signal *signal_code* to the process
feature(*s*) **from** *POSIX_CURRENT_PROCESS*
  -- POSIX locale specifics
set_native_messages
    -- Select native language as the language in which messages
    -- are displayed.
feature(*s*) **from** *PAPI_WAIT*
  -- C binding functions
posix_wait (*statloc*: *POINTER*): *INTEGER*
    -- Waits for process termination
posix_waitpid (*a_pid*: *INTEGER*; *statloc*: *POINTER*; *options*: *INTEGER*): *INTEGER*
    -- Waits for process termination
feature(*s*) **from** *PAPI_WAIT*
  -- C binding statloc evaluation
posix_wexitstatus (*a_value*: *INTEGER*): *INTEGER*
    -- Evaluates to the low-order eight bits of the *status*
    -- argument that the child passed to exit, or the value the
    -- child process returned from main.
posix_wifexited (*a_value*: *INTEGER*): *BOOLEAN*
    -- Evaluates to a non-zero value if status was returned for
    -- a child that terminated normally
posix_wifsignaled (*a_value*: *INTEGER*): *BOOLEAN*
    -- Evaluates to a non-zero value if status was returned for
    -- a child that terminated due to the receipt of a signal
    -- that was not caught
posix_wifstopped (*a_value*: *INTEGER*): *BOOLEAN*
posix_wstopsig (*a_value*: *INTEGER*): *BOOLEAN*
posix_wtermsig (*a_value*: *INTEGER*): *INTEGER*
feature(*s*) **from** *PAPI_WAIT*
  -- waitpid contants
wnohang: *INTEGER*
    -- do not suspend execution
wuntraced: *INTEGER*
    -- report status of childs that are stopped and whose status has not

    -- yet been reported since they stopped
feature(*s*) **from** *ABSTRACT_EXEC_PROCESS*
   -- (re)set arguments
   has_void_argument (*a_arguments*: *ARRAY*[*STRING*]): *BOOLEAN*
     -- Is one of the items in *a_arguments* Void?
   set_arguments (*a_arguments*: *ARRAY*[*STRING*])
feature(*s*) **from** *ABSTRACT_EXEC_PROCESS*
   -- i/o capturing
   capture_input: *BOOLEAN*
     -- is input captured on execute?
   capture_output: *BOOLEAN*
     -- is output captured on execute?
   capture_error: *BOOLEAN*
     -- is error captured on execute?
   set_capture_input (*on*: *BOOLEAN*)
   set_capture_output (*on*: *BOOLEAN*)
   set_capture_error (*on*: *BOOLEAN*)
   fd_stdin: *POSIX_FILE_DESCRIPTOR*
     -- Input read by process
   fd_stdout: *POSIX_FILE_DESCRIPTOR*
     -- Output emitted by process
   fd_stderr: *POSIX_FILE_DESCRIPTOR*
     -- Error output from process
feature(*s*) **from** *ABSTRACT_EXEC_PROCESS*
   -- Execute
   execute
     -- Execute *program_name* with arguments *arguments*. After
     -- execution, at some point in time, you have to *wait* or
     -- *wait_for* for this process to terminate.
     -- Current setting for error handling is retained for the
     -- captured i/o on the parent side, but not for the childs
     -- side (but maybe should??).
feature(*s*) **from** *ABSTRACT_EXEC_PROCESS*
   -- Access
   program_name: *STDC_PATH*
     -- Program to execute
   arguments: *ARRAY*[*STRING*]
     -- Arguments to pass to *program_name*
feature(*s*) **from** *POSIX_FORK_ROOT*
   -- termination info
   is_terminated_normally: *BOOLEAN*
     -- Has this process been terminated normally?
   is_exited: *BOOLEAN*
     -- Has this process been terminated normally?
   is_signalled: *BOOLEAN*
     -- Was child process terminated due to receipt of a signal
     -- that was not caught?
   signal_code: *INTEGER*
     -- Signal which caused the process to terminate

invariant

    --2007-12-13: invariant failure in some cases, root cause not determined yet

    --pid_known_is_not_terminated: is_pid_valid = not is_terminated

    accessing_real_singleton: *security_is_real_singleton*;

    valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;

    program_name_not_empty: *program_name* /= *Void* **and then not** *program_name.is_empty*;

    arguments_not_void: *arguments* /= *Void*;

    all_arguments_not_void: **not** *has_void_argument*(*arguments*);

    descriptors_are_owners: (*fd_stdin* /= *Void* **and then** *fd_stdin.is_open* **implies** *fd_stdin.is_owner*)
and then (*fd_stdout* /= *Void* **and then** *fd_stdout.is_open* **implies** *fd_stdout.is_owner*)
and then (*fd_stderr* /= *Void* **and then** *fd_stderr.is_open* **implies** *fd_stderr.is_owner*);

    streams_are_not_owner: (*stdin* /= *Void* **implies not** *stdin.is_owner*) **and then** (*stdout*
/= *Void* **implies not** *stdout.is_owner*) **and then** (*stderr* /= *Void* **implies not** *stderr.is_owner*);

end *of POSIX_EXEC_PROCESS*

C.9 Short form of POSIX_FILE

deferred class *interface POSIX_FILE*
feature(*s*) **from** *POSIX_FILE*
   -- special makes
   make_from_file_descriptor (*a_file_descriptor*: *ABSTRACT_FILE_DESCRIPTOR*;
a_mode: *STRING*)
      -- Open a stream from a given file descriptor.
      -- The stream will become leading so when the file
      -- descriptor is closed, it will not close, you have to close
      -- the stream to close the file descriptor.
      -- The stream will also inherit the error handling setting
      -- of a_file_descriptor.
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
   open_in_sync: *is_open_read* **or** *is_open_write* **implies** *is_open*; *-- The reverse is not*
true, for examples sockets can be
 -- closed for reading/writing, but still open.
   capacity_not_negative: *capacity* $>= 0$;
   valid_capacity: *is_open* $=$ (*capacity* > 0);
   open_implies_handle_assigned: *is_open* $=$ (*stream* $/=$ *unassigned_value*);
   owned_implies_open: *is_owner* **implies** *is_open*;
   owned_implies_handle_assigned: *is_owner* **implies** *stream* $/=$ *unassigned_value*;
   last_string_valid: *last_string* $/=$ *Void*;
   gets_buf_valid: *gets_buf* $/=$ *Void*;
end *of* **deferred** *POSIX_FILE*

C.10 Short form of POSIX_FILE_DESCRIPTOR

class *interface POSIX_FILE_DESCRIPTOR*
creation
  open (*a_path*: *STRING*; *a_flags*: *INTEGER*)
    -- Open given file with access given by *flags*.
  open_read (*a_path*: *STRING*)
    -- Open given file with access given by *flags*.
  open_write (*a_path*: *STRING*)
  open_read_write (*a_path*: *STRING*)
  open_truncate (*a_path*: *STRING*)
    -- Open file, if it exists, truncate it first.
  create_read_write (*a_path*: *STRING*)
    -- Always create a file, existing or not.
    -- Give read/write permissions to user only.
  create_write (*a_path*: *STRING*)
    -- Always create a file, existing or not.
    -- Give read/write permissions to user only.
  create_with_mode (*a_path*: *STRING*; *flags, mode*: *INTEGER*)
    -- Create a file according to *flags* and with *mode* access
    -- permissions. Make sure you have th O_CREAT flag in flags
    -- if you really want to create something!
  make_as_duplicate (*another*: *ABSTRACT_DESCRIPTOR*)
    -- On creation, create a duplicate from another descriptor.
    -- As normal call, closes its own descriptor first (if open) and
    -- duplicates next.
  make_from_file (*file*: *STDC_FILE*)
    -- Create file descriptor from given stream
    -- The stream is leading, so this file descriptor will
    -- never close itself, unless it is made an owner.
  attach_to_fd (*a_fd*: *INTEGER*; *a_become_owner*: *BOOLEAN*)
    -- Create descriptor with value *a_fd*. Descriptor will close
    -- it when *a_become_owner*.
feature(*s*) **from** *MEMORY*
  dispose
    -- Close handle if owner.
feature(*s*) **from** *KI_INPUT_STREAM*
  -- Input
  non_blocking_read_character
    -- Read the next item in input stream.
    -- Make the result available in *last_item*.
  non_blocking_read_to_buffer (*a_buffer*: *KI_BUFFER*[*CHARACTER*]; *pos, nb*: *INTEGER*):
INTEGER
    -- Fill *a_buffer*, starting at position *pos*, with
    -- at most *nb* items read from input stream.
    -- Return the number of items actually read.
feature(*s*) **from** *KI_INPUT_STREAM*
  -- Status report
  is_closable_for_reading: *BOOLEAN*

-- Can current input stream be closed?

is_open_read: *BOOLEAN*
-- Can items be read from input stream?

is_rewindable: *BOOLEAN*
-- Can current input stream be rewound to return input from
-- the beginning of the stream?

end_of_input: *BOOLEAN*
-- Has end-of-file been reached?

valid_unread_character (*a_character*: *CHARACTER*): *BOOLEAN*
-- Can *a_character* be put back in input stream?

feature(*s*) **from** *KI_INPUT_STREAM*

-- Access

name: *STDC_PATH*
-- Scratch path

last_character: *CHARACTER*
-- Last character read by *read_character* and a few other
-- routines

feature(*s*) **from** *KI_INPUT_STREAM*

-- Basic operations

close_for_reading
-- Try to close input stream if it is closable. Set
-- *is_open_read* to false if operation was successful.

rewind
-- Move input positionto the beginning of stream.

feature(*s*) **from** *KL_IMPORTED_ANY_ROUTINES*

-- Access

*any_*: *KL_ANY_ROUTINES*
-- Routines that ought to be in class ANY

feature(*s*) **from** *KI_CHARACTER_INPUT_STREAM*

-- Input

non_blocking_read_string (*nb*: *INTEGER*)
-- Read at most *nb* characters from input stream.
-- Make the characters that have actually been read
-- available in *last_string*.

non_blocking_read_to_string (*a_string*: *STRING*; *pos, nb*: *INTEGER*): *INTEGER*
-- Fill *a_string*, starting at position *pos*, with
-- at most *nb* characters read from input stream.
-- Return the number of characters actually read.

feature(*s*) **from** *KI_CHARACTER_INPUT_STREAM*

-- Access

last_string: *STRING*
-- Last string read;
-- (Note: this query always return the same object.
-- Therefore a clone should be used if the result
-- is to be kept beyond the next call to this feature.
-- However *last_string* is not shared between file objects.)

feature(*s*) **from** *EPX_CHARACTER_INPUT_STREAM*

-- Access

is_streaming: *BOOLEAN*

    -- Is data coming from a network stream?
feature(*s*) **from** *EPX_CHARACTER_INPUT_STREAM*
  -- Input
  last_read: *INTEGER*
    -- Last bytes read by *read_buffer*;
    -- Can be less than requested for non-blocking input.
    -- Check *last_blocked* in that case.
  read_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)
    -- Read data into *buf* at *offset* for *nbytes* bytes.
    -- Number of bytes actually read are available in *last_read*.
    -- This is a more safe version of *read* in case you have a
    -- STDC_BUFFER object.
feature(*s*) **from** *KI_TEXT_INPUT_STREAM*
  -- Input
  read_line
    -- Read characters from input stream until a line separator
    -- or end of file is reached. Make the characters that have
    -- been read available in *last_string* and discard the line
    -- separator characters from the input stream.
    -- Zero characters will be read when non-blocking i/o
    -- is enabled, and *read_line* would block at the first character.
    -- If a character has been read, *read_line* will block until
    -- a %N has been read or *end_of_input* occurs.
  read_new_line
    -- Read a line separator from input file.
    -- Make the characters making up the recognized
    -- line separator available in *last_string*,
    -- or make *last_string* empty and leave the
    -- input file unchanged if no line separator
    -- was found.
feature(*s*) **from** *KI_TEXT_INPUT_STREAM*
  -- Access
  eol: *STRING*
    -- Line separator;
    -- EPX classes do not distinguish between a %R%N or just %N
    -- end-of-line. The platform may though.
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
  -- The singleton, available to any because its used in preconditions
  security: *STDC_SECURITY*
    -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
  -- Access
  errno: *STDC_ERRNO*
    -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
  -- Status
  raise_exception_on_error: *BOOLEAN*
    -- Should an exception be raised when an error occurs?
    -- If not, you have to check errno for any errors.

feature(*s*) **from** *STDC_BASE*
  -- Change
  set_default_action_on_error
    -- Use *security.error_handling.exceptions_enabled* to
    -- determine if an exception should be raised when a C call
    -- returns an error.
  set_raise_exception_on_error
    -- Always raise an exception when a C call returns an error.
  set_continue_on_error
    -- Never raise an exception when a C call returns an error.
  inherit_error_handling (*an_instance*: *STDC_BASE*)
    -- Handle errors like *an_instance*
feature(*s*) **from** *KI_OUTPUT_STREAM*
  -- Output
  put_character (*c*: *CHARACTER*)
    -- Write a character.
  append (*an_input_stream*: *KI_INPUT_STREAM*[*CHARACTER*])
    -- Read items of *an_input_stream* until the end
    -- of input is reached, and write these items to
    -- current output stream.
    -- *append* is safe for non-blocking descriptors.
feature(*s*) **from** *KI_OUTPUT_STREAM*
  -- Status report
  is_open_write: *BOOLEAN*
    -- Can items be written to output stream?
  is_closable_for_writing: *BOOLEAN*
    -- Can current output stream be closed?
feature(*s*) **from** *KI_OUTPUT_STREAM*
  -- Basic operations
  close_for_writing
    -- Try to close output stream if it is closable. Set
    -- *is_open_write* to false if operation was successful.
feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
  -- Output
  put_string (*a_string*: *STRING*)
    -- Write *a_string* to output stream.
  put_substring (*a_string*: *STRING*; *s, e*: *INTEGER*)
    -- Write substring of *a_string* between indexes
    -- *s* and *e* to output stream.
  put_integer (*i*: *INTEGER*)
    -- Write decimal representation
    -- of *i* to output stream.
    -- Regexp: 0|(-?[1-9][0-9]*)
  put_boolean (*b*: *BOOLEAN*)
    -- Write "True" to output stream if
    -- *b* is true, "False" otherwise.
feature(*s*) **from** *KI_CHARACTER_OUTPUT_STREAM*
  -- Basic operations
  flush

   -- Flush buffered data to disk.

feature(*s*) **from** *EPX_CHARACTER_OUTPUT_STREAM*

 -- Output

 last_written: *INTEGER*

  -- How many bytes were written by the last call to a routine;

  -- Can be less than requested for non-blocking output.

  -- Check *last_blocked* in that case.

 put_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)

  -- More safe version of *write* in case you have a

  -- STDC_BUFFER object.

 write_buffer (*buf*: *STDC_BUFFER*; *offset, nbytes*: *INTEGER*)

  -- More safe version of *write* in case you have a

  -- STDC_BUFFER object.

feature(*s*) **from** *KI_TEXT_OUTPUT_STREAM*

 -- Output

 put_line (*a_string*: *STRING*)

  -- Write *a_string* to output stream

  -- followed by a line separator.

 put_new_line

  -- Write a line separator to output stream.

feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*

 -- Basic operations

 close

  -- Close the resource.

feature(*s*) **from** *EPX_CHARACTER_IO_STREAM*

 -- Status report

 is_closable: *BOOLEAN*

  -- Can current stream be closed for reading and writing?

 is_open: *BOOLEAN*

  -- Does *handle* contain an open handle?

 is_owner: *BOOLEAN*

  -- Does this object close the stream on *close* or *dispose*?

  -- Only for resources that are owned, are resource limits checked.

feature(*s*) **from** *STDC_HANDLE*

 -- Access

 resource_usage_can_be_increased: *BOOLEAN*

  -- Is it allowed to open another file?

feature(*s*) **from** *STDC_HANDLE*

 -- Influence ownership of the handle. Can help to influence subtile garbage collector problems

 become_owner

  -- This class will own its handle. This is the only function

  -- that actually increases the resource count.

 unown

  -- Resource will not be closed on dispose. Calling close will

  -- be forbidden. This routine may not call any other object,

  -- else it cannot be called from within dispose.

feature(*s*) **from** *STDC_HANDLE*

 -- Close

 detach

      -- Forget the resource. Resource is not closed.
      -- You cannot read and write anymore.
feature(*s*) **from** *STDC_HANDLE*
  -- Resource
  capacity: *INTEGER*
      -- Number of resources that are in use by *handle*. For a
      -- file this is 1, for a memory handle, this is the number of
      -- bytes.
  fd: *H*
      -- Identifier of resource tracked by this class.
feature(*s*) **from** *PORTABLE_PATH*
  -- Change
  set_portable_path (*a_path*: *STRING*)
      -- Set *portable_path* to *a_path*.
feature(*s*) **from** *HASHABLE*
  hash_code: *INTEGER*
      -- Hash code value
feature(*s*) **from** *STDC_HANDLE_BASED_IO_STREAM*
  -- Stream or disk file
  set_streaming (*enable*: *BOOLEAN*)
      -- Influence behaviour of certain functions if they should be
      -- optimized for data coming from disk or data coming from
      -- the network. In particular *is_streaming* implies that a
      -- client application is prepared to handle *read*s that
      -- return less than the requested number of bytes, but dont
      -- assume that means end-of-file.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Initialization
  make
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Special creation
  attach_to_fd (*a_fd*: *INTEGER*; *a_become_owner*: *BOOLEAN*)
      -- Create descriptor with value *a_fd*. Descriptor will close
      -- it when *a_become_owner*.
  make_as_duplicate (*another*: *ABSTRACT_DESCRIPTOR*)
      -- On creation, create a duplicate from another descriptor.
      -- As normal call, closes its own descriptor first (if open) and
      -- duplicates next.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Read and write to memory block
  last_blocked: *BOOLEAN*
      -- Would last call to *read* or *write* block?
  read (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
      -- Read data into *buf* at *offset* for *nbytes* bytes.
      -- The number of bytes actually read, is available in *last_read*.
  write (*buf*: *POINTER*; *offset, nbytes*: *INTEGER*)
      -- Write given data from *buf* at *offset*, for *nbytes*
      -- bytes. Number of actually written bytes are in
      -- *last_written. last_written* can be unequal to *nbytes*

    -- if i/o is non-blocking or some error has occurred.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Eiffel like output
  put (*a*: *ANY*)
    -- Write any Eiffel object as string using its *out* value.
  write_character (*c*: *CHARACTER*)
    -- Write a character.
  write_string (*a_string*: *STRING*)
    -- Write *a_string* to output stream.
  puts (*a_string*: *STRING*)
    -- Write *a_string* to output stream.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Buffered input
  read_character
    -- Sets *last_character*.
    -- If this routine blocks, *last_character* has the value
    -- %U. Therefore, if non-blocking is enabled, always check
    -- *last_blocked* to see if the value make sense.
  read_string (*nb*: *INTEGER*)
    -- Read at most *nb* characters from input stream.
    -- Make the characters that have actually been read
    -- available in *last_string*.
    -- Zero characters will be read when non-blocking i/o
    -- is enabled, and *read* would block.
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Status report
  is_attached_to_terminal: *BOOLEAN*
    -- Is the handle associated with character device?
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- Access
  value: *INTEGER*
    -- The actual file descriptor value
feature(*s*) **from** *ABSTRACT_DESCRIPTOR*
  -- non-blocking i/o
  is_blocking_io: *BOOLEAN*
    -- Is blocking i/o enabled (default)?
  set_blocking_io (*enable*: *BOOLEAN*)
    -- Set *is_blocking_io*.
  supports_nonblocking_io: *BOOLEAN*
    -- Does this descriptor support non-blocking input/output?
    -- On POSIX systems, any descriptor does.
    -- On Windows, sockets and pipes do.
feature(*s*) **from** *ABSTRACT_FILE_DESCRIPTOR*
  -- Initialization
  open (*a_path*: *STRING*; *a_flags*: *INTEGER*)
    -- Open given file with access given by *flags*.
  open_read (*a_path*: *STRING*)
    -- Open given file with access given by *flags*.
  open_write (*a_path*: *STRING*)

open_read_write (*a_path*: *STRING*)
open_truncate (*a_path*: *STRING*)
   -- Open file, if it exists, truncate it first.
create_read_write (*a_path*: *STRING*)
   -- Always create a file, existing or not.
   -- Give read/write permissions to user only.
create_write (*a_path*: *STRING*)
   -- Always create a file, existing or not.
   -- Give read/write permissions to user only.
create_with_mode (*a_path*: *STRING*; *flags, mode*: *INTEGER*)
   -- Create a file according to *flags* and with *mode* access
   -- permissions. Make sure you have th O_CREAT flag in flags
   -- if you really want to create something!
feature(*s*) **from** *ABSTRACT_FILE_DESCRIPTOR*
-- File position
seek (*offset*: *INTEGER*)
   -- Set file position to given absolute *offset.*
seek_from_current (*offset*: *INTEGER*)
   -- Set file position relative to current position.
seek_from_end (*offset*: *INTEGER*)
   -- Set file position relative to end of file.
feature(*s*) **from** *ABSTRACT_FILE_DESCRIPTOR*
-- Access
status: *POSIX_STATUS*
   -- The status for this file descriptor. Cached value,
   -- refreshed only when file reopened.
feature(*s*) **from** *PAPI_UNISTD*
-- C binding miscellaneous
posix_alarm (*a_seconds*: *INTEGER*): *INTEGER*
   -- Schedules an alarm.
posix_environ: *POINTER*
   -- The list of environment variables passed to this program
posix_execvp (*file*: *POINTER*; *argv*: *POINTER*): *INTEGER*
   -- Executes a program.
posix___exit (*a_status*: *INTEGER*)
   -- Cause program termination without calling exit handlers
   -- defined with *atexit.*
   -- *a_status* is returned to its parent.
posix_fork: *INTEGER*
   -- Create a process.
posix_getlogin: *POINTER*
   -- User name
posix_pause: *INTEGER*
   -- Wait for signal.
posix_sleep (*seconds*: *INTEGER*): *INTEGER*
   -- Delay process execution.
feature(*s*) **from** *PAPI_UNISTD*
-- functions with path as argument
posix_access (*a_path*: *POINTER*; *amode*: *INTEGER*): *INTEGER*

-- Tests for file accessibility
posix__chdir (*a__path*: *POINTER*): *INTEGER*
-- Changes the current working directory
posix__chown (*a__path*: *POINTER*; *a__owner, a__group*: *INTEGER*): *INTEGER*
-- Changes the owner and/or group of a file
posix__getcwd (*a__buf*: *POINTER*; *a__size*: *INTEGER*): *POINTER*
-- Gets current working directory
posix__link (*existing, new*: *POINTER*): *INTEGER*
-- Creates a link to a file
posix__rmdir (*a__path*: *POINTER*): *INTEGER*
-- Removes a directory
posix__unlink (*a__path*: *POINTER*): *INTEGER*
-- Removes a directory entry
feature(*s*) **from** *PAPI__UNISTD*
-- C binding file descriptor routines
posix__close (*fildes*: *INTEGER*): *INTEGER*
-- Closes a a file
posix__dup (*fildes*: *INTEGER*): *INTEGER*
-- Duplicates an open file descriptor
posix__dup2 (*fildes, fildes2*: *INTEGER*): *INTEGER*
-- Duplicates an open file descriptor
posix__fdatasync (*fildes*: *INTEGER*): *INTEGER*
-- Synchronize the data of a file
posix__fsync (*fildes*: *INTEGER*): *INTEGER*
-- Synchronize the state of a file
posix__fpathconf (*fildes*: *INTEGER*; *name*: *INTEGER*): *INTEGER*
-- Gets configuration variable for an open file
posix__isatty (*fildes*: *INTEGER*): *BOOLEAN*
-- Determines if a file descriptor is associated with a terminal
posix__lseek (*fildes*: *INTEGER*; *offset, whence*: *INTEGER*): *INTEGER*
-- Repositions read/write file offset
posix__pipe (*fildes*: *POINTER*): *INTEGER*
-- Creates an interprocess channel
posix__read (*fildes*: *INTEGER*; *buf*: *POINTER*; *nbyte*: *INTEGER*): *INTEGER*
-- Reads from a file
posix__ttyname (*fildes*: *INTEGER*): *POINTER*
-- Determines a terminal pathname
posix__write (*fildes*: *INTEGER*; *buf*: *POINTER*; *nbyte*: *INTEGER*): *INTEGER*
-- Reads from a file
feature(*s*) **from** *PAPI__UNISTD*
-- user and group ids
posix__getegid: *INTEGER*
-- Gets effective group ID
posix__geteuid: *INTEGER*
-- Gets effective user ID
posix__getgid: *INTEGER*
-- Gets real group ID
posix__getgroups (*gidsetsize*: *INTEGER*; *grouplist*: *POINTER*): *INTEGER*
-- Gets supplementary group IDs

posix_getpgrp: *INTEGER*
 -- Gets process group ID
posix_getpid: *INTEGER*
 -- Gets process ID
posix_getppid: *INTEGER*
 -- Gets parent process ID
posix_getuid: *INTEGER*
 -- Gets real user ID
posix_group_item (*a_grouplist*: *POINTER*; *a_item*: *INTEGER*): *INTEGER*
 -- Gets a gid from a list returned by *getgroups*
posix_setgid (*gid*: *INTEGER*): *INTEGER*
 -- Sets group ID
posix_setpgid (*pid, pgid*: *INTEGER*): *INTEGER*
 -- Sets process group ID for job control
posix_setsid: *INTEGER*
 -- Creates a session and sets the process group ID
posix_setuid (*uid*: *INTEGER*): *INTEGER*
 -- Sets user ID
feature(*s*) **from** *PAPI_UNISTD*
-- sysconf, note that -1 will be returned in case functionality is not supported
posix_sc_arg_max: *INTEGER*
 -- The length of the arguments for the exec() function
posix_sc_child_max: *INTEGER*
 -- The number of simultaneous processes per real user ID
posix_sc_clk_tck: *INTEGER*
 -- The number of clock ticks per second
posix_sc_ngroups_max: *INTEGER*
 -- The number of simultaneous supplementary group IDs
posix_sc_stream_max: *INTEGER*
 -- The maximum number of streams that one process can have
 -- open at one time.
posix_sc_tzname_max: *INTEGER*
 -- The maximum number of bytes in a timezone name.
posix_sc_open_max: *INTEGER*
 -- The maximum number of files that one process can have
 -- open at one time.
posix_sc_pagesize: *INTEGER*
 -- granularity in bytes of memory mapping and process memory locking
posix_sc_job_control: *INTEGER*
 -- Job control functions are supported.
posix_sc_saved_ids: *INTEGER*
 -- Each process has a saved set-user-ID and a saved set-group-ID
posix_sc_version: *INTEGER*
 -- Indicates the 4-digit year and 2-digit month that the
 -- standard was approved.
feature(*s*) **from** *PAPI_UNISTD*
-- capability constants
posix_asynchronous_io: *BOOLEAN*
 -- True if _POSIX_ASYNCHRONOUS_IO is defined

def__fsync: *BOOLEAN*
   -- True if __POSIX__FSYNC is defined
posix__mapped__files: *BOOLEAN*
   -- True if __POSIX__MAPPED__FILES is defined
posix__memlock: *BOOLEAN*
   -- True if __POSIX__MEMLOCK is defined
posix__memlock__range: *BOOLEAN*
   -- True if __POSIX__MEMLOCK__RANGE is defined
posix__memory__protection: *BOOLEAN*
   -- True if __POSIX__MEMORY__PROTECTION is defined
posix__message__passing: *BOOLEAN*
   -- True if __POSIX__MESSAGE__PASSING is defined
posix__priority__scheduling: *BOOLEAN*
   -- True if __POSIX__PRIORITY__SCHEDULING is defined
posix__semaphores: *BOOLEAN*
   -- True if __POSIX__SEMAPHORES is defined
posix__shared__memory__objects: *BOOLEAN*
   -- True if __POSIX__SHARED__MEMORY__OBJECTS is defined
def__synchronized__io: *BOOLEAN*
   -- True if __POSIX__SYNCHRONIZED__IO is defined
posix__timers: *BOOLEAN*
   -- True if __POSIX__TIMERS is defined
posix__threads: *BOOLEAN*
   -- True if __POSIX__THREADS is defined

feature(*s*) **from** *POSIX__FILE__DESCRIPTOR*
  -- Initialization
make__from__file (*file*: *STDC__FILE*)
   -- Create file descriptor from given stream
   -- The stream is leading, so this file descriptor will
   -- never close itself, unless it is made an owner.

feature(*s*) **from** *POSIX__FILE__DESCRIPTOR*
  -- Status
is__closed__on__execute: *BOOLEAN*
   -- Is this descriptor closed when the process executes or
   -- spawns a child process?

feature(*s*) **from** *POSIX__FILE__DESCRIPTOR*
  -- Close
close__on__execute
   -- Close this descriptor in the child process after a spawn
   -- or execute has happened. Important if you dont
   -- inadvertedly want to leak important sockets to a client.

feature(*s*) **from** *POSIX__FILE__DESCRIPTOR*
  -- Synchronisation
supports__file__synchronization: *BOOLEAN*
   -- Do we support synchronization?
supports__data__synchronization: *BOOLEAN*
   -- Do we support synchronization of data without metadata?
synchronize
   -- Synchronize the state of a file (includes synchronize__data).

synchronize_data
    -- Synchronize the data of a file. Cheaper than
    -- *synchronize*, but not always supported.
feature(*s*) **from** *POSIX_FILE_DESCRIPTOR*
  -- Locking
get_lock (*lock_to_test*: *POSIX_LOCK*): *POSIX_LOCK*
    -- Gets lock information, returns True if a lock is set on
    -- the region in a_lock. a_lock is overwritten with that lock.
set_lock_failed: *BOOLEAN*
    -- Did set_lock obtain a lock?
attempt_lock (*a_lock*: *POSIX_LOCK*)
    -- Attempt to set lock, if not possible, set
    -- *set_lock_failed*.
set_lock (*a_lock*: *POSIX_LOCK*)
    -- Attempt to set lock, wait if necessary.
feature(*s*) **from** *POSIX_FILE_DESCRIPTOR*
  -- Access
file_descriptor_flags: *INTEGER*
    -- All file descriptor bits associated with this handle.
terminal: *POSIX_TERMIOS*
    -- Terminal settings.
ttyname: *STRING*
    -- Terminal path name, or empty if this file descriptor does
    -- not refer to a terminal
invariant
accessing_real_singleton: *security_is_real_singleton*;
valid_error_action: *error_action* $>=$ *0* **and** *error_action* $<=$ *2*;
open_in_sync: *is_open_read* **or** *is_open_write* **implies** *is_open*; -- The reverse is not
true, for examples sockets can be
 -- closed for reading/writing, but still open.
capacity_not_negative: *capacity* $>=$ *0*;
valid_capacity: *is_open* $=$ (*capacity* $>$ *0*);
open_implies_handle_assigned: *is_open* $=$ (*fd* $/=$ *unassigned_value*);
owned_implies_open: *is_owner* **implies** *is_open*;
owned_implies_handle_assigned: *is_owner* **implies** *fd* $/=$ *unassigned_value*;
line_buffer_index_offset_ok: *line_buffer* $/=$ *Void* **implies** *line_buffer_index* $<=$ *line_buffer.count*;
valid_status: **not** *is_open* **implies** *my_status* $=$ *Void*;
end *of POSIX_FILE_DESCRIPTOR*

C.11 Short form of POSIX__FILE__SYSTEM

class *interface POSIX__FILE__SYSTEM*
feature(*s*) **from** *STDC__SECURITY__ACCESSOR*
   -- The singleton, available to any because its used in preconditions
   security: *STDC__SECURITY*
     -- Singleton entry point for security.
feature(*s*) **from** *STDC__BASE*
   -- Access
   errno: *STDC__ERRNO*
     -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC__BASE*
   -- Status
   raise__exception__on__error: *BOOLEAN*
     -- Should an exception be raised when an error occurs?
     -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC__BASE*
   -- Change
   set__default__action__on__error
     -- Use *security.error__handling.exceptions__enabled* to
     -- determine if an exception should be raised when a C call
     -- returns an error.
   set__raise__exception__on__error
     -- Always raise an exception when a C call returns an error.
   set__continue__on__error
     -- Never raise an exception when a C call returns an error.
   inherit__error__handling (*an__instance*: *STDC__BASE*)
     -- Handle errors like *an__instance*
feature(*s*) **from** *STDC__FILE__SYSTEM*
   -- Path names
   expand__path (*a__path*: *STRING*): *STDC__PATH*
     -- returns a new path
feature(*s*) **from** *STDC__FILE__SYSTEM*
   -- Rename files/directories, remove files/directories
   remove__file (*a__path*: *STRING*)
     -- calls *unlink* when *a__path* is a file, or *rmdir* when
     -- *a__path* is a directory.
     -- error when file could not be removed (and it exists)
   rename__to (*current__path, new__path*: *STRING*)
     -- Rename a file or a directory.
     -- *new__path* should not be an existing path.
feature(*s*) **from** *STDC__FILE__SYSTEM*
   -- Accessibility of files
   is__modifiable (*a__path*: *STRING*): *BOOLEAN*
     -- tests if file is readable and writable by this program
     -- uses real user ID and real group ID instead of effective ones
   is__readable (*a__path*: *STRING*): *BOOLEAN*
     -- Tests if *a__path* is readable by this program. *a__path*
     -- can be a file or a directory.

-- Uses real user ID and real group ID instead of effective
-- ones.
feature(*s*) **from** *STDC_FILE_SYSTEM*
-- File and string
write_string_to_file (*s, a_file_name*: *STRING*)
-- Write *s* to file *a_file_name.*
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
-- Directory access
change_directory (*a_directory*: *STRING*)
-- Changes the current working directory.
current_directory: *STRING*
-- The current directory
make_directory (*a_directory*: *STRING*)
-- Makes a directory, only accessible by owner.
mkdir (*a_directory*: *STRING*)
-- Makes a directory, only accessible by owner.
remove_directory (*a_directory*: *STRING*)
-- Removes an empty directory, does not fail if directory
-- does not exist
rmdir (*a_directory*: *STRING*)
-- Removes an empty directory, does not fail if directory
-- does not exist
force_remove_directory (*a_directory*: *STRING*)
-- Removes a directory, even when not empty.
-- I suggest you do not have hard or symbolic links in *a_directory...*
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
-- File statistics
status (*a_path*: *STRING*): *POSIX_STATUS_PATH*
-- Gets information about a file
status_may_fail (*a_path*: *STRING*): *ABSTRACT_STATUS_PATH*
-- Retrieve status information for *a_path. a_path* may or
-- may not exist. Check *Result.found* to see if statistics
-- were retrieved.
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
-- Directory browsing
browse_directory (*a_path*: *STRING*): *POSIX_DIRECTORY*
-- Get information about a directory.
find_program_in_path (*a_filename*: *STRING*; *a_paths*: *ARRAY*[*STRING*]): *STRING*
-- Look for *a_filename* in *a_paths,* check if it is a
-- binary and return the full path to *a_filename* when
-- found. Return Void if not found.
feature(*s*) **from** *ABSTRACT_FILE_SYSTEM*
-- Accessibility of files
last_access_result: *INTEGER*
-- value of last access test
is_accessible (*a_path*: *STRING*; *a_mode*: *INTEGER*): *BOOLEAN*
-- Is *a_path* accessibility using *a_mode*?
access (*a_path*: *STRING*; *a_mode*: *INTEGER*): *BOOLEAN*
-- Is *a_path* accessibility using *a_mode*?

is__directory (*a__path*: *STRING*): *BOOLEAN*
   -- Does *a__path* exists and is it a directory?
is__existing (*a__path*: *STRING*): *BOOLEAN*
   -- Is *a__path* an existing file, directory, whatever?
   -- Tests if file does exist, not if it is readable or writable by
   -- this program!
   -- Uses real user ID and real group ID instead of effective ones.
is__empty (*a__path*: *STRING*): *BOOLEAN*
   -- True if file exists and has a size equal to zero.
is__executable (*a__path*: *STRING*): *BOOLEAN*
   -- tests if file is executable by this program
is__regular__file (*a__path*: *STRING*): *BOOLEAN*
   -- Does *a__path* exists and is it a regular file?
is__writable (*a__path*: *STRING*): *BOOLEAN*
   -- tests if file is writable by this program
   -- uses real user ID and real group ID instead of effective ones

feature(*s*) **from** *ABSTRACT__FILE__SYSTEM*
-- File system properties
is__case__sensitive: *BOOLEAN*
   -- is file system case sensitive or not?
path__separator: *CHARACTER*
   -- What is the path separator?

feature(*s*) **from** *ABSTRACT__FILE__SYSTEM*
-- File and string
file__content__as__string (*a__file__name*: *STRING*): *STRING*
   -- Contents of *a__file__name* as a STRING

feature(*s*) **from** *ABSTRACT__FILE__SYSTEM*
-- Path names
resolved__path__name (*a__path*: *STRING*): *STRING*
   -- Absolute pathname derived from *a__path* that names the
   -- same file, whose resolution does not involve ".", "..", or
   -- symbolic links
temporary__directory: *STRING*
   -- the temporary directory

feature(*s*) **from** *POSIX__FILE__SYSTEM*
-- Read/write permissions
chmod (*a__path*: *STRING*; *a__mode*: *INTEGER*)
   -- Changes file mode for *a__path* to *a__mode*.
change__mode (*a__path*: *STRING*; *a__mode*: *INTEGER*)
   -- Changes file mode for *a__path* to *a__mode*.
permissions (*a__path*: *STRING*): *POSIX__PERMISSIONS*
   -- The permissions object (a new one every time!) for the
   -- given file
set__read__only (*a__path*: *STRING*)
   -- Make given file read__only.
set__writable (*a__path*: *STRING*)
   -- Make given *a__path* read__only.

feature(*s*) **from** *POSIX__FILE__SYSTEM*
-- File times

touch (*a_path*: *STRING*)
    -- Sets the modification and access times of *a_path* to the
    -- current time of day.
    -- File is created if it does not exist.
utime (*a_path*: *STRING*; *access_time, modification_time*: *POSIX_TIME*)
    -- Sets file access and modification times.
feature(*s*) **from** *POSIX_FILE_SYSTEM*
  -- Further directory access
link (*existing, new*: *STRING*)
    -- Create a hard link to a file.
unlink (*a_path*: *STRING*)
    -- Remove a directory entry, should be a file, not a directory.
    -- Its not an error if path does not exist, but all other
    -- errors are reported.
feature(*s*) **from** *POSIX_FILE_SYSTEM*
  -- FIFOs
create_fifo (*a_path*: *STRING*; *a_mode*: *INTEGER*)
    -- Create a FIFO special file.
mkfifo (*a_path*: *STRING*; *a_mode*: *INTEGER*)
    -- Create a FIFO special file.
feature(*s*) **from** *POSIX_FILE_SYSTEM*
  -- Shared memory
unlink_shared_memory_object (*name*: *STRING*)
    -- Remove a shared memory object.
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
end *of POSIX_FILE_SYSTEM*

C.12   Short form of POSIX_FORK_ROOT

deferred class *interface POSIX_FORK_ROOT*
feature(*s*) **from** *STDC_CHILD_PROCESS*
  -- Termination info
  has_exit_code: *BOOLEAN*
    -- Does *exit_code* return a valid value?
  is_terminated: *BOOLEAN*
    -- Is child not running any more?
  exit_code: *INTEGER*
    -- Low-order 8 bits of call to _exit or exit for this process
feature(*s*) **from** *ABSTRACT_CHILD_PROCESS*
  -- Actions that parent may execute
  wait_for (*suspend*: *BOOLEAN*)
    -- Wait for this process to terminate. If *suspend* then we
    -- wait until the information about this process is available,
    -- else we return immediately.
    -- If suspend is False, check the running property to see
    -- if this child is really terminated.
feature(*s*) **from** *ARGUMENTS*
  command_name: *STRING*
feature(*s*) **from** *CAPI_TIME*
  -- Standard C binding
  current_time: *INTEGER*
    -- The current calendar time in seconds since the epoch
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Process standard input/output/error
  stdin: *POSIX_TEXT_FILE*
  stdout: *POSIX_TEXT_FILE*
  stderr: *POSIX_TEXT_FILE*
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Various
  clock: *INTEGER*
    -- Approximation of processor time used by the program, or -1
    -- if unknown
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Random numbers
  random: *INTEGER*
    -- Returns a pseudo-random integer between 0 and *RAND_MAX*.
  set_random_seed (*a_seed*: *INTEGER*)
    -- Sets *a_seed* as the seed for a new sequence of
    -- pseudo-random integers to be returned by *random*. These
    -- sequences are repeatable by calling *set_random_seed* with
    -- the same seed value. If no seed value is provided, the
    -- *random* function is automatically seeded with a value of
    -- 1.
feature(*s*) **from** *STDC_CURRENT_PROCESS*
  -- Global locale
  locale: *STRING*

-- Current locale
numeric_format: *STDC_LOCALE_NUMERIC*
   -- Various information for formatting numbers and monetary
   -- quantities
set_locale (*category*: *INTEGER*; *new_locale*: *STRING*)
   -- Set given locale to *new_locale*. *new_locale* is either a
   -- well-known constant like "C" or "da_DK" or an opaque
   -- string that was returned by another call of *setlocale*.
set_c_locale
   -- Set locale to the Standard C locale (the default).
set_native_decimal_point
   -- Set the decimal point character using the LC_NUMERIC
   -- environment variable.
set_native_locale
   -- Set entire locale to the natives setting which is
   -- determend by environment variables like LC_NUMERIC,
   -- LC_COLLATE, LC_CTYPE etc.
set_native_time
   -- Set time display to the natives setting using the LC_TIME
   -- environment variable.
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- Access
effective_user_name: *STRING*
   -- Name of the user currently associated with the current
   -- thread;
   -- Name will not be Void, but can be empty if no name found
   -- (you can screw up your /etc/passwd on Unix...)
full_command_name: *STRING*
   -- *command_name* with fully qualified path;
   -- An empty string is returned in case *command_name* is
   -- empty. As any program can setup the arguments passed to
   -- another program, an empty *command_name* is a possibility.
child_pid: *INTEGER*
   -- The process identifier.
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- Status
is_child_pid_valid: *BOOLEAN*
   -- return True if this object refers to a child process, so
   -- it has an id
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- Every process also has standard file descriptors which might not be compatible with
stdin/stdout/stderr (Windows)
 fd_stdin: *POSIX_FILE_DESCRIPTOR*
 fd_stdout: *POSIX_FILE_DESCRIPTOR*
 fd_stderr: *POSIX_FILE_DESCRIPTOR*
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- Sleeping
millisleep (*a_milliseconds*: *INTEGER*)
   -- Sleep for *a_milliseconds* milliseconds. Due to timer

        -- resolution issues, the minimum resolution might be in the
        -- order of 10ms or higher.
   sleep (*seconds*: *INTEGER*)
        -- Delays process execution up to *seconds*. Can return early
        -- if interrupted. Check *unslect_seconds*
   unslept_seconds: *INTEGER*
        -- The number of seconds still to sleep, before being
        -- interrupted; it is set by *sleep*. If it is zero, no
        -- interrupt occurred and process slept for the allotted
        -- time.
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
  -- The singleton, available to any because its used in preconditions
  security: *STDC_SECURITY*
        -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
  -- Access
  errno: *STDC_ERRNO*
        -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
  -- Status
  raise_exception_on_error: *BOOLEAN*
        -- Should an exception be raised when an error occurs?
        -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
  -- Change
  set_default_action_on_error
        -- Use *security.error_handling.exceptions_enabled* to
        -- determine if an exception should be raised when a C call
        -- returns an error.
  set_raise_exception_on_error
        -- Always raise an exception when a C call returns an error.
  set_continue_on_error
        -- Never raise an exception when a C call returns an error.
  inherit_error_handling (*an_instance*: *STDC_BASE*)
        -- Handle errors like *an_instance*
feature(*s*) **from** *ABSTRACT_PROCESS*
  -- Signal this process
  terminate
        -- attempt to gracefully terminate this process
feature(*s*) **from** *EPX_CURRENT_PROCESS*
  -- Access (doesnt make a lot of sense if youre not inheriting)
  raw_environment_variables: *ARRAY*[*STRING*]
        -- The raw list of name=value pairs of environment
        -- variables passed to this process;
        -- A new list is created every time this feature is accessed.
feature(*s*) **from** *POSIX_PROCESS*
  -- signal this process
  kill (*a_signal_code*: *INTEGER*)
        -- Send signal *signal_code* to the process

feature(*s*) **from** *POSIX_CURRENT_PROCESS*
  -- POSIX locale specifics
  set_native_messages
    -- Select native language as the language in which messages
    -- are displayed.
feature(*s*) **from** *PAPI_WAIT*
  -- C binding functions
  posix_wait (*statloc*: *POINTER*): *INTEGER*
    -- Waits for process termination
  posix_waitpid (*a_pid*: *INTEGER*; *statloc*: *POINTER*; *options*: *INTEGER*): *INTEGER*
    -- Waits for process termination
feature(*s*) **from** *PAPI_WAIT*
  -- C binding statloc evaluation
  posix_wexitstatus (*a_value*: *INTEGER*): *INTEGER*
    -- Evaluates to the low-order eight bits of the *status*
    -- argument that the child passed to exit, or the value the
    -- child process returned from main.
  posix_wifexited (*a_value*: *INTEGER*): *BOOLEAN*
    -- Evaluates to a non-zero value if status was returned for
    -- a child that terminated normally
  posix_wifsignaled (*a_value*: *INTEGER*): *BOOLEAN*
    -- Evaluates to a non-zero value if status was returned for
    -- a child that terminated due to the receipt of a signal
    -- that was not caught
  posix_wifstopped (*a_value*: *INTEGER*): *BOOLEAN*
  posix_wstopsig (*a_value*: *INTEGER*): *BOOLEAN*
  posix_wtermsig (*a_value*: *INTEGER*): *INTEGER*
feature(*s*) **from** *PAPI_WAIT*
  -- waitpid contants
  wnohang: *INTEGER*
    -- do not suspend execution
  wuntraced: *INTEGER*
    -- report status of childs that are stopped and whose status has not
    -- yet been reported since they stopped
feature(*s*) **from** *POSIX_FORK_ROOT*
  -- termination info
  is_terminated_normally: *BOOLEAN*
    -- Has this process been terminated normally?
  is_exited: *BOOLEAN*
    -- Has this process been terminated normally?
  is_signalled: *BOOLEAN*
    -- Was child process terminated due to receipt of a signal
    -- that was not caught?
  signal_code: *INTEGER*
    -- Signal which caused the process to terminate
invariant
  --2007-12-13: invariant failure in some cases, root cause not determined yet
  --pid_known_is_not_terminated: is_pid_valid = not is_terminated
  accessing_real_singleton: *security_is_real_singleton*;

valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
end *of* **deferred** *POSIX_FORK_ROOT*

C.13 Short form of POSIX_GROUP

class *interface POSIX_GROUP*
creation
   make_from_name (*a_name*: *STRING*)
   make_from_gid (*a_gid*: *INTEGER*)
feature(*s*) **from** *POSIX_GROUP*
   -- creation
   make_from_name (*a_name*: *STRING*)
   make_from_gid (*a_gid*: *INTEGER*)
feature(*s*) **from** *POSIX_GROUP*
   -- refresh cache
   refresh
      -- refresh cache with latest info from user database
feature(*s*) **from** *POSIX_GROUP*
   -- queries
   name: *STRING*
      -- group name
   gid: *INTEGER*
      -- ID number
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
   valid_group: *group* $/=$ *default_pointer*;
end *of POSIX_GROUP*

C.14 Short form of POSIX_LOCK

class *interface POSIX_LOCK*
creation
  make
feature(*s*) **from** *POSIX_LOCK*
  -- creation
  make
feature(*s*) **from** *POSIX_LOCK*
  -- members
  allow_read: *BOOLEAN*
    -- This is a read lock
  allow_all: *BOOLEAN*
    -- No lock or used to remove a lock
  allow_none: *BOOLEAN*
    -- This is a write lock
  start: *INTEGER*
  length: *INTEGER*
  pid: *INTEGER*
feature(*s*) **from** *POSIX_LOCK*
  -- settable members
  set_allow_read
    -- this is a read or shared lock
  set_allow_all
    -- to remove a lock
  set_allow_none
    -- this is a write or exclusive lock
  set_seek_start
    -- start is measured from the beginning of the file
  set_seek_current
    -- start is measured from the current position
  set_seek_end
    -- start is measured from the end of the file
  set_start (*a_start*: *INTEGER*)
    -- set relative offset in bytes
  set_length (*a_length*: *INTEGER*)
    -- number of bytes to lock
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
  valid_buf: *buf* /= *Void*;
  lock_type_known: *allow_all* **or else** *allow_none* **or else** *allow_read*;
end *of POSIX_LOCK*

C.15 Short form of POSIX_MEMORY_MAP

class *interface POSIX_MEMORY_MAP*
creation
  make (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset, a_size*: *INTEGER*; *a_base*: *POINTER*;
a_prot, a_flags: *INTEGER*)
    -- Raw interface to mmap.
    -- This function can fail on certain system (Linux for
    -- example) if a_offset is not a multiple of PAGE_SIZE.
  make_private (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset, a_size*: *INTEGER*)
    -- Make the given file descriptor. *a_fd* should have been opened
    -- with read/write access.
    -- This is a mapping where changes are private.
    -- *a_offset* denotes the offset from *a_fd*.
    -- This function can fail on certain system (Linux for
    -- example) if a_offset is not a multiple of PAGE_SIZE.
  make_shared (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset, a_size*: *INTEGER*)
    -- Make the given file descriptor. *a_fd* should have been opened
    -- with read/write access.
    -- This is a mapping where changes are shared, i.e. the
    -- *a_offset* denotes the offset from *a_fd*.
    -- underlying object is also changed.
    -- This function can fail on certain system (Linux for
    -- example) if a_offset is not a multiple of PAGE_SIZE.
feature(*s*) **from** *POSIX_MEMORY_MAP*
  -- Initialization
  make (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset, a_size*: *INTEGER*; *a_base*: *POINTER*;
a_prot, a_flags: *INTEGER*)
    -- Raw interface to mmap.
    -- This function can fail on certain system (Linux for
    -- example) if a_offset is not a multiple of PAGE_SIZE.
  make_private (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset, a_size*: *INTEGER*)
    -- Make the given file descriptor. *a_fd* should have been opened
    -- with read/write access.
    -- This is a mapping where changes are private.
    -- *a_offset* denotes the offset from *a_fd*.
    -- This function can fail on certain system (Linux for
    -- example) if a_offset is not a multiple of PAGE_SIZE.
  make_shared (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset, a_size*: *INTEGER*)
    -- Make the given file descriptor. *a_fd* should have been opened
    -- with read/write access.
    -- This is a mapping where changes are shared, i.e. the
    -- *a_offset* denotes the offset from *a_fd*.
    -- underlying object is also changed.
    -- This function can fail on certain system (Linux for
    -- example) if a_offset is not a multiple of PAGE_SIZE.
feature(*s*) **from** *POSIX_MEMORY_MAP*
  -- Unmap
  close

    -- Remove the mapping.
feature(*s*) **from** *POSIX_MEMORY_MAP*
  -- Access
  fd: *POSIX_FILE_DESCRIPTOR*
    -- The file that is mapped.
  offset: *INTEGER*
    -- Offset in *fd* where mapping begins.
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
  capacity_not_negative: *capacity* >= *0*;
  valid_capacity: *is_allocated* = (*capacity* > *0*);
  open_implies_handle_assigned: *is_allocated* = (*ptr* /= *unassigned_value*);
  owned_implies_open: *is_owner* **implies** *is_allocated*;
  owned_implies_handle_assigned: *is_owner* **implies** *ptr* /= *unassigned_value*;
  size_positive: *is_open* **implies** *capacity* > *0*;
  ptr_valid: *is_open* **implies** *ptr* /= *default_pointer* **and not** *is_open* **implies** *ptr* =
default_pointer;
  offset_not_negative: *offset* >= *0*;
  have_file_descriptor: *fd* /= *Void*;
  file_descriptor_open: *fd.is_open*;
end *of POSIX_MEMORY_MAP*

C.16 Short form of POSIX_PERMISSIONS

deferred class *interface POSIX_PERMISSIONS*
feature(*s*) **from** *POSIX_PERMISSIONS*
  apply
    -- make permissions changes (if any) permanent
  refresh
    -- synchronize with permission changes possibly made on disk
feature(*s*) **from** *POSIX_PERMISSIONS*
  -- query mode
  allow_anyone_execute: *BOOLEAN*
    -- anyone allowed to execute the file?
  allow_anyone_read: *BOOLEAN*
    -- anyone allowed to read the file?
  allow_anyone_read_write: *BOOLEAN*
    -- anyone allowed to read and write the file?
  allow_anyone_write: *BOOLEAN*
    -- anyone allowed to write the file?
  allow_group_execute: *BOOLEAN*
    -- process with a group ID that matches the files group
    -- allowed to execute the file?
  allow_group_read: *BOOLEAN*
    -- process with a group ID that matches the files group
    -- allowed to read the file?
  allow_group_read_write: *BOOLEAN*
    -- process with a group ID that matches the files group
    -- allowed to read the file?
  allow_group_write: *BOOLEAN*
    -- process with a group ID that matches the files group
    -- allowed to write the file?
  allow_owner_execute: *BOOLEAN*
    -- owner allowed to execute the file
  allow_read: *BOOLEAN*
  allow_owner_read: *BOOLEAN*
  allow_read_write: *BOOLEAN*
  allow_owner_read_write: *BOOLEAN*
  allow_write: *BOOLEAN*
  allow_owner_write: *BOOLEAN*
  is_set_group_id: *BOOLEAN*
    -- group ID set on execution?
  is_set_gid: *BOOLEAN*
    -- group ID set on execution?
  is_set_user_id: *BOOLEAN*
    -- user ID set on execution?
  is_set_uid: *BOOLEAN*
    -- user ID set on execution?
feature(*s*) **from** *POSIX_PERMISSIONS*
  -- set permissions
  set_allow_anyone_execute (*allow*: *BOOLEAN*)

      -- give anyone execute permission
    set_allow_anyone_read (*allow*: *BOOLEAN*)
      -- give anyone read permission
    set_allow_anyone_read_write (*allow*: *BOOLEAN*)
      -- give anyone read and write permissions
    set_allow_anyone_write (*allow*: *BOOLEAN*)
      -- give anyone write permission
    set_allow_group_execute (*allow*: *BOOLEAN*)
      -- give group execute permission
    set_allow_group_read (*allow*: *BOOLEAN*)
      -- give group read permission
    set_allow_group_read_write (*allow*: *BOOLEAN*)
      -- give group read and write permission
    set_allow_group_write (*allow*: *BOOLEAN*)
      -- give group write permission
    set_allow_owner_execute (*allow*: *BOOLEAN*)
      -- give owner execute permission
    set_allow_read (*allow*: *BOOLEAN*)
      -- give read permission
    set_allow_owner_read (*allow*: *BOOLEAN*)
      -- give read permission
    set_allow_read_write (*allow*: *BOOLEAN*)
      -- give read/write permission
    set_allow_write (*allow*: *BOOLEAN*)
      -- give write permission
    set_allow_owner_write (*allow*: *BOOLEAN*)
      -- give write permission
  feature(*s*) **from** *POSIX_PERMISSIONS*
    -- direct access to Unix fields
    uid: *INTEGER*
      -- id of object owner, always 0 on NT
    owner_id: *INTEGER*
      -- id of object owner, always 0 on NT
    gid: *INTEGER*
      -- id of group, always 0 on NT
    group_id: *INTEGER*
      -- id of group, always 0 on NT
    mode: *INTEGER*
      -- the bit coded Unix mode field
  feature(*s*) **from** *POSIX_PERMISSIONS*
    -- set owner and group
    set_owner_id (*a_owner_id*: *INTEGER*)
      -- change the owner
    set_group_id (*a_group_id*: *INTEGER*)
      -- change the group
  invariant
    accessing_real_singleton: *security_is_real_singleton*;
    valid_error_action: *error_action $>= 0$* **and** *error_action $<= 2$*;
  end *of* **deferred** *POSIX_PERMISSIONS*

C.17 Short form of POSIX_PIPE

```
class interface POSIX_PIPE
creation
   make
      -- Create pipe.
feature(s) from POSIX_PIPE
   -- the pipe
   fdin: POSIX_FILE_DESCRIPTOR
      -- Incoming end of pipe
   fdout: POSIX_FILE_DESCRIPTOR
      -- Outgoing end of pipe
invariant
   accessing_real_singleton: security_is_real_singleton;
   valid_error_action: error_action >= 0 and error_action <= 2;
   valid_pipe: fdin /= Void and fdout /= Void;
end of POSIX_PIPE
```

C.18   Short form of POSIX_SEMAPHORE

deferred class *interface POSIX_SEMAPHORE*
invariant
    accessing_real_singleton: *security_is_real_singleton*;
    valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
    capacity_not_negative: *capacity* >= *0*;
    valid_capacity: *is_open* = (*capacity* > *0*);
    open_implies_handle_assigned: *is_open* = (*handle* /= *unassigned_value*);
    owned_implies_open: *is_owner* **implies** *is_open*;
    owned_implies_handle_assigned: *is_owner* **implies** *handle* /= *unassigned_value*;
end *of* **deferred** *POSIX_SEMAPHORE*

C.19 Short form of POSIX_SIGNAL

class *interface POSIX_SIGNAL*
creation
  make (*a_value*: *INTEGER*)
feature(*s*) **from** *POSIX_SIGNAL*
  -- Initialization
  make (*a_value*: *INTEGER*)
feature(*s*) **from** *POSIX_SIGNAL*
  -- Set signal properties, make effective with *apply*
  apply
    -- Make changes effective.
  set_child_stop (*stop*: *BOOLEAN*)
    -- Generate SIGCHLD when children stop.
  set_default_action
    -- Install signal-specific default action when *apply* is called.
  set_ignore_action
    -- Ignore signal when *apply* is called..
  set_handler (*a_handler*: *STDC_SIGNAL_HANDLER*)
    -- Install ones own signal handler when *apply* is called.
  set_mask (*a_mask*: *POSIX_SIGNAL_SET*)
feature(*s*) **from** *POSIX_SIGNAL*
  -- signal functions
  raise_in (*a_pid*: *INTEGER*)
    -- Raise the signal in the given process.
feature(*s*) **from** *POSIX_SIGNAL*
  -- Signal state
  child_stop: *BOOLEAN*
    -- generate SIGCHLD when children stop
  handler: *POINTER*
    -- pointer to function which catches this signal
  is_defaulted: *BOOLEAN*
    -- signal is handled by its specific default action
  is_ignored: *BOOLEAN*
    -- signal is ignored
  is_ignorable: *BOOLEAN*
    -- True if this signal is ignorable, either it is so by
    -- default or it may be set so.
  mask: *POSIX_SIGNAL_SET*
  refresh
    -- get latest state for this signal
invariant
  --accessing_real_singleton: signal_switch_is_real_singleton
  -- Gives crash with ISE Eiffel
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
  valid_signal_value: *value* $>= 1$;
  has_memory: *sigaction* $/=$ *Void*;
end *of POSIX_SIGNAL*

C.20   Short form of POSIX_SIGNAL_SET

class *interface POSIX_SIGNAL_SET*
creation
   make_empty
     -- make an initially empty signal set
   make_full
     -- make a set where all signals are enabled
   make_pending
     -- this signal set will be the set of signals that are blocked
     -- and pending
feature(*s*) **from** *POSIX_SIGNAL_SET*
   -- creation, make a set
   make_empty
     -- make an initially empty signal set
   make_full
     -- make a set where all signals are enabled
   make_pending
     -- this signal set will be the set of signals that are blocked
     -- and pending
feature(*s*) **from** *POSIX_SIGNAL_SET*
   -- change a set
   extend (*signo*: *INTEGER*)
     -- add signal to set
   put (*signo*: *INTEGER*)
     -- add signal to set
   prune (*signo*: *INTEGER*)
     -- remove the signal from the set
   wipe_out
     -- remove all items
feature(*s*) **from** *POSIX_SIGNAL_SET*
   -- commands to do something with set
   add_to_blocked_signals
     -- Add the signals to the set of blocked signals
   remove_from_blocked_signals
     -- Remove the signals from the set of blocked signals
   set_blocked_signals
     -- Set the set of blocked signals to this set
   suspend
     -- Suspend process, until delivery of a signal whose action
     -- is either to execute a signal-catching function or to
     -- terminate the process
feature(*s*) **from** *POSIX_SIGNAL_SET*
   -- queries
   has (*signo*: *INTEGER*): *BOOLEAN*
     -- is signal *signo* in the set
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;

```
    have_set: set /= Void;
  end of POSIX_SIGNAL_SET
```

C.21 Short form of POSIX_STATUS

deferred class *interface POSIX_STATUS*
feature(*s*) **from** *POSIX_STATUS*
   -- stat members
   is_block_special: *BOOLEAN*
     -- True if block-special file
   ino: *INTEGER*
   inode: *INTEGER*
   permissions: *POSIX_PERMISSIONS*
     -- file permissions
    ensure
      valid_result: *Result /= Void*
feature(*s*) **from** *POSIX_STATUS*
   -- direct access to the unix fields, not recommended
   unix_gid: *INTEGER*
   unix_uid: *INTEGER*
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action >= 0* **and** *error_action <= 2*;
   stat_not_void: *stat /= Void* **and then** *stat.capacity >= abstract_stat_size*;
end *of* **deferred** *POSIX_STATUS*

C.22 Short form of POSIX_SYSTEM

class *interface POSIX_SYSTEM*
feature(*s*) **from** *POSIX_SYSTEM*
    -- Sysconf queries, run-time determined
    child_max: *INTEGER*
      -- The number of simultaneous processes per real user ID.
    clock_ticks: *INTEGER*
      -- The number of clock ticks per second.
    has_job_control: *BOOLEAN*
      -- Job control functions are supported.
    has_saved_ids: *BOOLEAN*
      -- Each process has a saved set-user-ID and a saved set-group-ID.
    ngroups_max: *INTEGER*
      -- The number of simultaneous supplementary group IDs.
    page_size: *INTEGER*
      -- granularity in bytes of memory mapping and process memory locking.
    posix_version: *INTEGER*
      -- Indicates the 4-digit year and 2-digit month that the
      -- standard was approved.
feature(*s*) **from** *POSIX_SYSTEM*
    -- Compile-time determined queries
    supports_asynchronous_io: *BOOLEAN*
      -- True if the message passing API is supported.
    supports_file_synchronization: *BOOLEAN*
      -- True if file synchronization is supported.
    supports_memory_mapped_files: *BOOLEAN*
      -- True if memory mapped files are supported.
    supports_memory_locking: *BOOLEAN*
      -- True if memory locking is supported.
    supports_memlock_range: *BOOLEAN*
      -- True if memory range locking is supported.
    supports_memory_protection: *BOOLEAN*
      -- True if memory protection is supported.
    supports_message_passing: *BOOLEAN*
      -- True if the message passing API is supported.
    supports_priority_scheduling: *BOOLEAN*
      -- True if priority scheduling is supported.
    supports_semaphores: *BOOLEAN*
      -- True if semaphores are supported.
    supports_shared_memory_objects: *BOOLEAN*
      -- True if shared memory objects are supported.
    supports_synchronized_io: *BOOLEAN*
      -- True if synchronized io is supported.
    supports_timers: *BOOLEAN*
      -- True if timers are supported.
    supports_threads: *BOOLEAN*
      -- True if thread are supported.
invariant

     accessing_real_singleton: *security_is_real_singleton*;
     valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
  end *of POSIX_SYSTEM*

C.23 Short form of POSIX_TERMIOS

class *interface POSIX_TERMIOS*
creation
   make (*a_fd*: *POSIX_FILE_DESCRIPTOR*)
feature(*s*) **from** *POSIX_TERMIOS*
   -- Access, raw individual fields
   iflag: *INTEGER*
     -- Input mode flags
   oflag: *INTEGER*
     -- output mode flags
   cflag: *INTEGER*
     -- control mode flags
   lflag: *INTEGER*
     -- local mode flags
feature(*s*) **from** *POSIX_TERMIOS*
   -- More friendly settings
   is_input_echoed: *BOOLEAN*
     -- are input characters echoed back to the terminal?
   is_receiving: *BOOLEAN*
     -- If false, no characters are received
   set_echo_input (*enable*: *BOOLEAN*)
   set_echo_new_line (*enable*: *BOOLEAN*)
   set_input_control (*enable*: *BOOLEAN*)
     -- enable start/stop input control
   set_receive (*enable*: *BOOLEAN*)
feature(*s*) **from** *POSIX_TERMIOS*
   -- line control functions
   flush_input
     -- Discards all data that has been received but not read.
   drain
     -- Wait for all output to be transmitted to the terminal.
   send_break
     -- sends a break to the terminal
feature(*s*) **from** *POSIX_TERMIOS*
   -- Get/set baudrates as symbols
   input_speed: *INTEGER*
     -- The terminal input baud rate as symbolic value.
   output_speed: *INTEGER*
     -- The terminal output baud rate as symbolic value.
   set_input_speed (*new_rate*: *INTEGER*)
     -- Set terminal input baud rate, *new_rate* is one of the
     -- BXXXX constants
   set_output_speed (*new_rate*: *INTEGER*)
     -- Set terminal output baud rate, *new_rate* is one of the
     -- BXXXX constants
feature(*s*) **from** *POSIX_TERMIOS*
   -- symbol to baud rate conversions
   speed_to_baud_rate (*symbol*: *INTEGER*): *INTEGER*

-- Given a baud rate symbol, the real baud rate is returned.
feature(*s*) **from** *POSIX__TERMIOS*
  -- Apply/refresh state
  apply__now
    -- Change occurs immediately.
  apply__drain
    -- Change occurs after all output written to *fd* has been
    -- transmitted. This function should be used when changing
    -- parameters that affect output.
  apply__flush
    -- Change occurs after all output written to *fd* has been
    -- transmitted. All input that has been received but not
    -- read, is discarded before the change is made.
  refresh
    -- Get terminal settings currently in effect.
feature(*s*) **from** *POSIX__TERMIOS*
  -- Access
  fd: *POSIX__FILE__DESCRIPTOR*
    -- The file descriptor for these terminal settings.
invariant
  accessing__real__singleton: *security__is__real__singleton*;
  valid__error__action: *error__action* >= *0* **and** *error__action* <= *2*;
  valid__attr: *attr* /= *Void* **and then** *attr.capacity* = *posix__termios__size*;
  valid__fd: *fd* /= *Void*;
end *of POSIX__TERMIOS*

C.24 Short form of POSIX_TIMED_COMMAND

deferred class *interface POSIX_TIMED_COMMAND*
feature(*s*) **from** *POSIX_TIMED_COMMAND*
   -- Initialization
   make (*a_seconds*: *INTEGER*)
feature(*s*) **from** *POSIX_TIMED_COMMAND*
   -- Execution
   execute: *BOOLEAN*
     -- Did *do_execute* complete its task within *seconds* seconds?
feature(*s*) **from** *POSIX_TIMED_COMMAND*
   -- Access
   is_signal_alarm_handled: *BOOLEAN*
     -- Does the signal SIGNAL_ALARM cause an Eiffel exception?
feature(*s*) **from** *POSIX_TIMED_COMMAND*
   -- State
   remaining_seconds: *INTEGER*
     -- number of seconds left in previous request
   seconds: *INTEGER*
     -- the number of seconds available to execute the command
   set_seconds (*a_seconds*: *INTEGER*)
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
   valid_seconds: *seconds* $>= 1$ **and** *seconds* $<= 65535$;
end *of* **deferred** *POSIX_TIMED_COMMAND*

C.25 Short form of POSIX__USER

class *interface POSIX_USER*
creation
  make_from_name (a_name: STRING)
  make_from_uid (a_uid: INTEGER)
feature(*s*) **from** *POSIX_USER*
  -- creation
  make_from_name (a_name: STRING)
  make_from_uid (a_uid: INTEGER)
feature(*s*) **from** *POSIX_USER*
  -- Base commands
  refresh
    -- Refresh cache with latest info from user database.
feature(*s*) **from** *POSIX_USER*
  -- Access
  name: *STRING*
    -- login name
  uid: *INTEGER*
    -- ID number
  gid: *INTEGER*
    -- group ID number
  home_directory: *STRING*
    -- initial working directory
  shell: *STRING*
    -- initial user program
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
  valid_passwd: *passwd* $/=$ *default_pointer*;
end *of POSIX_USER*

C.26 Short form of POSIX_USER_DATABASE

class *interface POSIX_USER_DATABASE*
feature(*s*) **from** *POSIX_USER_DATABASE*
   -- Access
   is_existing_uid (*uid*: *INTEGER*): *BOOLEAN*
     -- Returns True if this uid exists in /etc/passwd
     -- (or through NIS or whatever mechanisms that might be in use)
   is_existing_login (*login*: *STRING*): *BOOLEAN*
     -- Returns True if this login exists in /etc/passwd
     -- (or through NIS or whatever mechanisms that might be in use)
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
end *of POSIX_USER_DATABASE*

D
Short (flat) listing of Single Unix Specification classes

Classes in this appendix are based on the Single Unix Specification. They inherit from the POSIX classes. Inherited features are not shown.

D.1 Short form of SUS_CONSTANTS

class *interface SUS_CONSTANTS*
feature(*s*) **from** *SUS_CONSTANTS*
  -- Syslog facility codes
  log_kern: *INTEGER*
    -- kernel messages
  log_user: *INTEGER*
    -- random user-level messages
  log_mail: *INTEGER*
    -- mail system
  log_daemon: *INTEGER*
    -- system daemons
  log_auth: *INTEGER*
    -- security/authorization messages
  log_lpr: *INTEGER*
    -- line printer subsystem
  log_news: *INTEGER*
    -- network news subsystem
  log_uucp: *INTEGER*
    -- UUCP subsystem
  log_cron: *INTEGER*
    -- clock daemon
  log_local0: *INTEGER*
    -- Reserved for local use
  log_local1: *INTEGER*
    -- Reserved for local use
  log_local2: *INTEGER*
    -- Reserved for local use
  log_local3: *INTEGER*

    -- Reserved for local use
log_local4: *INTEGER*
    -- Reserved for local use
log_local5: *INTEGER*
    -- Reserved for local use
log_local6: *INTEGER*
    -- Reserved for local use
log_local7: *INTEGER*
    -- Reserved for local use
feature(*s*) **from** *SUS_CONSTANTS*
  -- Syslog open options
log_pid: *INTEGER*
    -- log the pid with each message
log_cons: *INTEGER*
    -- log on the console if errors in sending
log_odelay: *INTEGER*
    -- delay open until first syslog() (default)
log_ndelay: *INTEGER*
    -- dont delay open
feature(*s*) **from** *SUS_CONSTANTS*
  -- Syslog priorities
log_emerg: *INTEGER*
log_alert: *INTEGER*
log_crit: *INTEGER*
log_err: *INTEGER*
log_warning: *INTEGER*
log_notice: *INTEGER*
log_info: *INTEGER*
log_debug: *INTEGER*
feature(*s*) **from** *SUS_CONSTANTS*
  -- Socket kinds
sock_dgram: *INTEGER*
    -- Connectionless, unreliable datagrams of fixed maximum length.
sock_packet: *INTEGER*
    -- Linux specific way of getting packets at the dev level.
    -- For writing rarp and other similar things on the user
    -- level.
sock_raw: *INTEGER*
    -- Raw protocol interface.
sock_seqpacket: *INTEGER*
    -- Sequenced, reliable, connection-based, datagrams of fixed
    -- maximum length.
sock_stream: *INTEGER*
    -- Sequenced, reliable, connection-based byte streams.
feature(*s*) **from** *SUS_CONSTANTS*
  -- Protocol families
af_inet: *INTEGER*
    -- Internet domain sockets for use with IPv4 addresses.
af_inet6: *INTEGER*

      -- Internet domain sockets for use with IPv6 addresses.
    af_unix: *INTEGER*
      -- UNIX domain sockets.
    af_unspec: *INTEGER*
      -- Unspecified.
feature(*s*) **from** *SUS_CONSTANTS*
  -- Shutdown options
  shut_rd: *INTEGER*
    -- No more receptions.
  shut_rdwr: *INTEGER*
    -- No more receptions or transmissions.
  shut_wr: *INTEGER*
    -- No more transmissions.
feature(*s*) **from** *SUS_CONSTANTS*
  -- h_errno values
  try_again: *INTEGER*
    -- Non-Authoritative Host not found, or SERVERFAIL.
  no_recovery: *INTEGER*
    -- Non recoverable errors, FORMERR, REFUSED, NOTIMP.
  no_data: *INTEGER*
    -- Valid name, no data record of requested type.
  no_address: *INTEGER*
    -- No address, look for MX record. Equal to NO_DATA.
feature(*s*) **from** *SUS_CONSTANTS*
  -- Lengths of string forms of ip addresses
  inet_addrstrlen: *INTEGER*
    -- Length of an IPv4 string.
  inet6_addrstrlen: *INTEGER*
    -- Length of an IPv6 string.
feature(*s*) **from** *SUS_CONSTANTS*
  -- Other constants
  somaxconn: *INTEGER*
    -- Maximum backlog.
feature(*s*) **from** *SUS_CONSTANTS*
  -- Socket options level for *getsockopt* and *setsockopt*
  sol_socket: *INTEGER*
  ipproto_ip: *INTEGER*
  ipproto_ipv6: *INTEGER*
  ipproto_icmp: *INTEGER*
  ipproto_icmpv6: *INTEGER*
  ipproto_raw: *INTEGER*
  ipproto_tcp: *INTEGER*
  ipproto_udp: *INTEGER*
feature(*s*) **from** *SUS_CONSTANTS*
  -- SOL_SOCKET option names
  so_rcvbuf: *INTEGER*
    -- Receive buffer size;
    -- 0 if option not supported (only on BeOS).
  so_reuseaddr: *INTEGER*

        -- Allow local address reuse
    so_sndbuf: *INTEGER*
        -- Send buffer size;
        -- 0 if option not supported (only on BeOS).
  feature(*s*) **from** *SUS_CONSTANTS*
    -- IP type-of-service options
    ip_tos: *INTEGER*
    iptos_lowdelay: *INTEGER*
    iptos_throughput: *INTEGER*
  feature(*s*) **from** *SUS_CONSTANTS*
    -- TCP options
    tcp_nodelay: *INTEGER*
  feature(*s*) **from** *SUS_CONSTANTS*
    -- Special IPv4 addresses
    inaddr_any: *INTEGER*
        -- 0.0.0.0
    inaddr_broadcast: *INTEGER*
        -- 255.255.255.255
    inaddr_loopback: *INTEGER*
        -- 127.0.0.1
  feature(*s*) **from** *SUS_CONSTANTS*
    -- Available clocks (-1 if not available)
    clock_realtime: *INTEGER*
    clock_monotonic: *INTEGER*
    clock_process_cputime_id: *INTEGER*
    clock_thread_cputime_id: *INTEGER*
  end *of SUS_CONSTANTS*

D.2 Short form of SUS_ENV_VAR

class *interface SUS_ENV_VAR*
creation
   make (a_name: STRING)
feature(*s*) **from** *SUS_ENV_VAR*
   -- Commands
   set_value (a_new_value: STRING)
      -- Change environment value. Repeatedly creating a new
      -- SUS_ENV_VAR and calling *set_value* will lead to a memory
      -- leak.
invariant
   accessing_real_singleton: security_is_real_singleton;
   valid_error_action: error_action >= 0 **and** *error_action <= 2;*
end *of SUS_ENV_VAR*

D.3 Short form of SUS_FILE_SYSTEM

class *interface SUS_FILE_SYSTEM*
feature(*s*) **from** *SUS_FILE_SYSTEM*
  -- File statistics
  status (*a_path*: *STRING*): *SUS_STATUS_PATH*
    -- Return information about path.
  symbolic_link_status (*a_path*: *STRING*): *SUS_STATUS*
    -- Return information about path, but if it is a symbolic
    -- link, about the symbolic link instead of the referenced path
feature(*s*) **from** *SUS_FILE_SYSTEM*
  -- Symbolic links
  create_symbolic_link (*old_path, new_path*: *STRING*)
    -- Creates a symbolic link
  symlink (*old_path, new_path*: *STRING*)
    -- Creates a symbolic link
feature(*s*) **from** *SUS_FILE_SYSTEM*
  -- File system properties
  resolved_path_name (*a_path*: *STRING*): *STRING*
    -- Derives from *a_path* an absolute pathname that names the
    -- same file, whose resolution does not involve ".", "..", or
    -- symbolic links.
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action >= 0* **and** *error_action <= 2*;
end *of SUS_FILE_SYSTEM*

D.4   Short form of SUS_HOST

class *interface SUS_HOST*
creation
   make_from_name (*a_name*: *STRING*)
     -- Initialize host from *name*. If *name* is numerical, the
     -- behaviour is not specified.
   make_from_address (*an_address*: *ABSTRACT_IP_ADDRESS*)
     -- Initialize host from ip address *an_address*.
     -- An attempt is made to resolve the host name using this address.
     -- Status is always found, even when reverse lookup failed.
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* $>= 0$ **and** *error_action* $<= 2$;
   name_void_or_not_empty: *name* $=$ *Void* **or else not** *name.is_empty*;
   has_canonical_name: *found* **implies** *name* $/=$ *Void* $=$ (*canonical_name* $/=$ *Void*);
   has_at_least_one_ip_address: *found* $=$ (*addresses* $/=$ *Void* **and then** *addresses.count*
> 0);
   only_non_void_addresses: *found* **implies** *is_every_address_not_void*;
   has_aliases: *found* $=$ (*aliases* $/=$ *Void*);
   valid_length: *found* **implies** *address_length* > 0;
   consistent: *addresses* $/=$ *Void* **and then** *addresses.count* > 0 **implies** *found*;
   my_not_found_reason_valid: *found* $=$ (*my_not_found_reason* $= 0$);
end *of SUS_HOST*

D.5 Short form of SUS_SERVICE

class *interface SUS_SERVICE*
creation
   make_from_name (*a_name, a_protocol*: *STRING*)
      -- Retrieve service information with *a_name* and optional
      -- *a_protocol* from services database.
      -- If service not found, an exception is raised.
   make_from_port (*a_port*: *INTEGER*; *a_protocol*: *STRING*)
      -- Initialize service from given a_port.
      -- Make sure to provide a *a_protocol* if necessary!
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
   name_void_or_not_empty: *name* = *Void* **or else** **not** *name.is_empty*;
   valid_port: *port* >= *0* **and** *port* <= *65535*;
   valid_protocol: *protocol* = *Void* **or else** *protocol.is_empty* **or else** (*protocol.is_equal*(*once_tcp*)
or *protocol.is_equal*(*once_udp*));
   valid_protocol_type: *protocol_type* = *sock_stream* **or else** *protocol_type* = *sock_dgram*;
   valid_aliases: *aliases* /= *Void*;
end *of SUS_SERVICE*

D.6 Short form of SUS_SOCKET_ADDRESS

class *interface SUS_SOCKET_ADDRESS*
"Use EPX_HOST_PORT instead."
end *of SUS_SOCKET_ADDRESS*

D.7 Short form of SUS_SYSLOG

class *interface SUS_SYSLOG*
feature(*s*) **from** *SUS_SYSLOG*
   -- open and close
   open (*a_identification*: *STRING*; *a_format, a_facility*: *INTEGER*)
     -- start logging with the given identification
   close
     -- stop logging
feature(*s*) **from** *SUS_SYSLOG*
   -- Write log messages, will auto-open if not is_open
   emergency (*msg*: *STRING*)
     -- the system is unusable
   alert (*msg*: *STRING*)
     -- action must be taken immediately
   critical (*msg*: *STRING*)
     -- critical conditions
   error (*msg*: *STRING*)
     -- error conditions
   warning (*msg*: *STRING*)
     -- warning conditions
   notice (*msg*: *STRING*)
     -- normal but significant condition
   info (*msg*: *STRING*)
     -- informational
   debug_dump (*msg*: *STRING*)
     -- Debug-level messages.
feature(*s*) **from** *SUS_SYSLOG*
   -- Access
   identification: *STRING*
   format: *INTEGER*
   facility: *INTEGER*
   is_open: *BOOLEAN*
invariant
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
   remain_single: *Current* = *singleton*;
   have_identification: *is_open* **implies** *identification* /= *Void* **and then not** *identification.is_empty*;
end *of SUS_SYSLOG*

D.8   Short form of SUS_TCP_SOCKET

class *interface SUS_TCP_SOCKET*
creation
    attach_to_socket (*a_fd*: *INTEGER*; *a_become_owner*: *BOOLEAN*)
      -- Create descriptor with value *a_fd*. Descriptor will close
      -- it when *a_become_owner*.
invariant
    accessing_real_singleton: *security_is_real_singleton*;
    valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
    open_in_sync: *is_open_read* **or** *is_open_write* **implies** *is_open*; -- The reverse is not
true, for examples sockets can be
 -- closed for reading/writing, but still open.
    capacity_not_negative: *capacity* >= *0*;
    valid_capacity: *is_open* = (*capacity* > *0*);
    open_implies_handle_assigned: *is_open* = (*fd* /= *unassigned_value*);
    owned_implies_open: *is_owner* **implies** *is_open*;
    owned_implies_handle_assigned: *is_owner* **implies** *fd* /= *unassigned_value*;
    line_buffer_index_offset_ok: *line_buffer* /= *Void* **implies** *line_buffer_index* <= *line_buffer.count*;
    unassigned_value_is_error_value: *unassigned_value* = *-1*;
end *of SUS_TCP_SOCKET*

Classes in this appendix are based on Standard C only.

E.1 Short form of EPX_CGI

```
deferred class interface EPX_CGI
feature(s) from EPX_CGI
   -- Output
   execute
      -- Execute the CGI action by emiting a valid MIME header and
      -- an optional body.
      -- Header and/or body text can be accumulated in
      -- as_uc_string and will be send to the client when this
      -- feature returns.
      -- In case of binary output it is advised to write the header
      -- yourself (build it up in header and use finish_header
      -- to write it, and write the binary output straight to
      -- stdout.
      require
         is_valid_request_method
feature(s) from EPX_CGI
   -- Error handling
   error_unauthorized
      -- Signal authorization error to client.
   error_invalid_method
      -- Signal invalid method to client.
feature(s) from EPX_CGI
   -- Debug support
   dump_input
      -- Write cgi input to $TMPDIR/cgi_input.
      -- First line contains the content header, is not actually in input!
feature(s) from EPX_CGI
   -- Status
   is_authorized: BOOLEAN
      -- May request_method be applied this resource?
      -- This method may implement any kind of authentication.
```

is_delete_method: *BOOLEAN*
    -- Is *request_method* equal to "DELETE"?
is_get_method: *BOOLEAN*
    -- Is *request_method* equal to "GET"?
is_head_method: *BOOLEAN*
    -- Is *request_method* equal to "GET"?
is_post_method: *BOOLEAN*
    -- Is *request_method* equal to "POST"?
is_put_method: *BOOLEAN*
    -- Is *request_method* equal to "PUT"?
is_http_header_written: *BOOLEAN*
    -- Has header been written to stdout?
    -- Such an action cannot be undone, and no more headers can
    -- be written.
is_valid_key (*a_key*: *STRING*): *BOOLEAN*
    -- Is *a_key* a valid key for *value* or *raw_value*?
is_valid_request_method: *BOOLEAN*
    -- Is *request_method* valid for this CGI?
feature(*s*) **from** *EPX_CGI*
  -- Access
header: *EPX_MIME_CGI_HEADER*
    -- Response header
feature(*s*) **from** *EPX_CGI*
  -- Standard CGI variables
auth_type: *STRING*
    -- Type of authentication used
content_type: *STRING*
    -- MIME type of data when invoked with POST method
content_length: *INTEGER*
    -- Length, in bytes, of data when invoked with POST method
gateway_interface: *STRING*
    -- Name and version of the gateway, for example CGI/1.1
http_accept: *STRING*
    -- Contents of the Accept header line sent by the client
http_cookie: *STRING*
    -- All cookies sent by the client in the form of key=value,
    -- semi-colon separated
http_referer: *STRING*
    -- Contents of the Referer header line
http_user_agent: *STRING*
    -- Name of the client program that is making the request
path_info: *STRING*
    -- Extra path information as it was passed to the server in
    -- the query URL
path_translated: *STRING*
    -- Extra path information translated to a final, usable
    -- form; the Web document root is prepended to the query
    -- path, and any other path translations are executed.
query_string: *STRING*

    -- The input when invoked with the GET method
remote_address: *STRING*
    -- IP address of the client that made the request
remote_host: *STRING*
    -- Name of the remote computer that made the request
remote_ident: *STRING*
    -- User name as given by the ident protocol
remote_user: *STRING*
    -- Name of the remote user that made the request
request_method: *STRING*
    -- Name of the method used to invoke the CGI application
remapped_request_method: *STRING*
    -- As *request_method* but if method remapping is enabled,
    -- return the remapped method
script_name: *STRING*
    -- Name of script that was invoked
server_name: *STRING*
    -- Domain name of the computer that is running the server software
server_port: *INTEGER*
    -- TCP port number on which the server that invoked the CGI
    -- application is operating
server_protocol: *STRING*
    -- Name of the protocol that the server is using and the
    -- version of that protocol. The protocol name and version
    -- are separated by a forward slash with no spaces, for
    -- instance HTTP/1.0
server_software: *STRING*
    -- Name of the server that is handling the request
feature(*s*) **from** *EPX_CGI*
  -- HTTP headers
if_match: *STRING*
    -- The contents of the If-Match header if set or if
    -- made available by the server;
    -- Void otherwise
    -- Bugs: If-Match: * not handled, has to be done manually.
if_none_match: *STRING*
    -- The contents of the If-None-Match header if set or if
    -- made available by the server;
    -- Void otherwise
    -- Bugs: If-None-Match: * not handled, has to be done manually.
if_modified_since: *STDC_TIME*
    -- The contents of the If-Modified-Since header if set or if
    -- made available by the server;
    -- Void otherwise
feature(*s*) **from** *EPX_CGI*
  -- CGI headers
content_text_html
    -- Write Content-Type: text/html to stdout.
    -- Clients will guess the charset, so its better to use the

-- explicit *context_text_html_utf8* function.
content_text_html_utf8
    -- Write Content-Type: text/html with explicit character ste
    -- UTF-8 to stdout.
    -- Use this when emitting UTF-8.
content_text_plain
    -- Write Content-Type: text/plain to stdout.
finish_header
    -- Finish the header by emitting an empty line.
    -- If *cookies* have been set, they are written as well.
location (*a_url*: *STRING*)
    -- Redirect to *a_url* by emitting a Location header.
    -- This is used to specify to the server that you are
    -- returning a reference to a document rather than an actual
    -- document.
    -- If the argument to this is a URL, the server will issue a
    -- redirect to the client.
    -- If the argument to this is a virtual path, the server will
    -- retrieve the document specified as if the client had
    -- requested that document originally. ? directives will work
    -- in here, but # directives must be redirected back to the
    -- client.
    -- If you return a status as well, it must be 200 it seems.
status (*a_status_code*: *INTEGER*; *a_reason*: *STRING*)
    -- Set the status code sent back to the client.
    -- This is used to give the server an HTTP/1.0 status line to
    -- send to the client. The format is nnn xxxxx, where nnn is
    -- the 3-digit status code, and xxxxx is the reason string,
    -- such as "Forbidden".
    -- Leave *a_reason* empty to return the default reason.
feature(*s*) **from** *EPX_CGI*
  -- Cookies
cookies: *DS_HASH_TABLE*[*EPX_HTTP_COOKIE,STRING*]
    -- Cookies that will be returned to the browser
set_cookie (*a_cookie*: *EPX_HTTP_COOKIE*)
    -- Add a new cookie that will be send to the browser then
    -- *context_text_html* is called.
feature(*s*) **from** *EPX_CGI*
  -- Server push, multipart header
content_multipart_x_mixed_replace (*boundary*: *STRING*)
    -- Initiate server push.
content_next_part
    -- Write boundary so next part of multipart msg can be written.
content_multipart_end
    -- Write boundary of multipart.
is_multipart_message: *BOOLEAN*
    -- Are we writing server push, multipart output?
feature(*s*) **from** *EPX_CGI*
  -- Form input

has_input: *BOOLEAN*
    -- Is input passed to cgi program?
has_key (*a_key*: *STRING*): *BOOLEAN*
    -- Is *a_key* passed as parameter/form-data?
is_meta_char (*c*: *CHARACTER*): *BOOLEAN*
    -- Is *c* a commonly used meta character?
meta_chars: *STRING*
    -- Commonly used meta characters.
    -- BdB: Check if this list is complete...
new_key_value_cursor (*a_key_re, a_value_re*: *RX_PCRE_REGULAR_EXPRESSION*;
an_on_match_found: *EPX_KEY_VALUE_MATCH*): *EPX_CGI_KEY_VALUE_CURSOR*
    -- New cursor to iterate over all keys, optionally including
    -- those keys and/or values that match *a_key_re* and
    -- *a_value_re;*
    -- Useful when a form returns table like names and you want
    -- to iterate over the keys for that table.
raw_value (*a_key*: *STRING*): *STRING*
    -- Returns value for key.
    -- if key does not exist, the empty string is returned.
remove_meta_chars (*s*: *STRING*)
    -- If *s* contains meta characters, theyre removed.
value (*a_key*: *STRING*): *STRING*
    -- As *raw_value* but meta characters are removed
invariant
    -- lower_a_code_definition: lower_a_code = (a).code
    -- Same thing for all other codes.
    -- (see "note" in indexing clause.)
    accessing_real_singleton: *security_is_real_singleton*;
    valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
    my_xml_not_void: *my_xml* /= *Void*;
    same_size: *attributes.count = values.count*;
    has_tag_stack: *tags* /= *Void*;
    comparing_references_is_not_good_enough: *tags.equality_tester* /= *Void*;
    fragment_has_no_header: *is_fragment* **implies** *is_header_written*;
    values_not_void: *values* /= *Void*;
    attributes_not_void: *attributes* /= *Void*;
    every_attribute_has_a_value: *attributes.count = values.count*;
    tag_state_valid: *tag_state* >= *tag_pending* **and** *tag_state* <= *tag_closed*;
    tag_started_and_pending_in_sync: *tag_state = tag_pending* **implies** *is_tag_started*;
    tag_closed_is_not_started: *tag_state = tag_closed = **not** is_tag_started*;
end *of* **deferred** *EPX_CGI*

E.2 Short form of EPX_MIME_PARSER

class *interface EPX_MIME_PARSER*
creation
  make
    -- Create a new parser.
  make_from_file (*a_file*: *STDC_TEXT_FILE*)
    -- Like *make_from_stream*, but turns off buffering in
    -- *a_file*.
  make_from_stream (*a_stream*: *EPX_CHARACTER_INPUT_STREAM*)
    -- Initialize parser, and set the input buffer to *a_stream*.
  make_from_string (*s*: *STRING*)
    -- Initialize parser, and set the input buffer to *s*.
  make_from_file_descriptor (*a_fd*: *ABSTRACT_FILE_DESCRIPTOR*)
feature(*s*) **from** *EPX_MIME_PARSER*
  -- Initialization
  make
    -- Create a new parser.
  make_from_file (*a_file*: *STDC_TEXT_FILE*)
    -- Like *make_from_stream*, but turns off buffering in
    -- *a_file*.
  make_from_stream (*a_stream*: *EPX_CHARACTER_INPUT_STREAM*)
    -- Initialize parser, and set the input buffer to *a_stream*.
  make_from_string (*s*: *STRING*)
    -- Initialize parser, and set the input buffer to *s*.
feature(*s*) **from** *EPX_MIME_PARSER*
  -- Character reading
  end_of_input: *BOOLEAN*
    -- Has *read_character* hit the end-of-file character?
feature(*s*) **from** *EPX_MIME_PARSER*
  -- Parsing
  reset_parsing_errors
    -- Reset count of parsing errors.
  parse
    -- Read input and build *part*.
    -- Check *syntax_error* for parsing errors.
  parse_body
    -- Parse MIME body.
    -- Assume *input_buffer* points to body part.
    -- If *a_content_length* positive, scans only as much body as
    -- given by *a_content_length*, given that the input buffer
    -- is an EPX_MIME_BUFFER.
  parse_header
    -- Read just the MIME header from the input and build a new
    -- *part*. Check *syntax_error* for parsing errors.
  set_header (*a_header*: *STRING*)
    -- Optional header that is parsed before the regular input
    -- is parsed.
  parsing_errors: *INTEGER*

-- Number of errors encountered when parsing.
feature(*s*) **from** *EPX_MIME_PARSER*
   -- Access
   read_first_body_part
     -- First part of the body, if any, if *parse_header* has been
     -- used;
     -- Even if only the header is parsed using *parse_header*,
     -- the first part of the body is still read by the parsers
     -- buffer as it doesnt know its part of the body at that
     -- time. Use this to retrieve the first part of the body, the
     -- rest of the body can be read from the stream the usual
     -- way.
   part: *EPX_MIME_PART*
     -- Structure were building
invariant
   -- lower_a_code_definition: lower_a_code = (a).code
   -- Same thing for all other codes.
   -- (see "note" in indexing clause.)
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
   yyss_not_void: *yyss* /= *Void*;
   yytranslate_not_void: *yytranslate* /= *Void*;
   yyr1_not_void: *yyr1* /= *Void*;
   yytypes1_not_void: *yytypes1* /= *Void*;
   yytypes2_not_void: *yytypes2* /= *Void*;
   yydefact_not_void: *yydefact* /= *Void*;
   yydefgoto_not_void: *yydefgoto* /= *Void*;
   yypact_not_void: *yypact* /= *Void*;
   yypgoto_not_void: *yypgoto* /= *Void*;
   yytable_not_void: *yytable* /= *Void*;
   yycheck_not_void: *yycheck* /= *Void*;
   input_buffer_not_void: *input_buffer* /= *Void*;
   valid_start_condition: *valid_start_condition*(*start_condition*);
   yy_content_not_void: *yy_content* /= *Void*;
   yy_line_positive: *yy_line* >= *1*;
   yy_column_positive: *yy_column* >= *1*;
   yy_position_positive: *yy_position* >= *1*;
   yy_nxt_not_void: *yy_nxt* /= *Void*;
   yy_chk_not_void: *yy_chk* /= *Void*;
   yy_base_not_void: *yy_base* /= *Void*;
   yy_def_not_void: *yy_def* /= *Void*;
   yy_accept_not_void: *yy_accept* /= *Void*;
   yy_state_stack_not_void: *yyreject_or_variable_trail_context* **implies** *yy_state_stack*
/= *Void*;
   never_a_keyword_in_start_condition: *start_condition* = *initial* **implies not** *expect_keyword*
and not *expect_keyword_after_space*;
   encoded_word_scanner_not_void: *encoded_word_scanner* /= *Void*;
   last_line_not_void: *last_line* /= *Void*;
   my_date_not_void: *my_date* /= *Void*;

```
    my_date_in_utc: my_date.is_utc_time;
    my_time_not_void: my_time /= Void;
    my_time_in_utc: my_time.is_utc_time;
end of EPX_MIME_PARSER
```

E.3   Short form of EPX_MIME_PART

class *interface EPX_MIME_PART*
creation
   make_empty
     -- Make an empty MIME part. Useful during parsing to
     -- construct the message gradually.
feature(*s*) **from** *EPX_MIME_PART*
   -- Output
   append_to_string (*s*: *STRING*)
     -- Serialize contents of MIME structure to *s*.
     -- Adds the Content-Length field with the proper length, if
     -- this field does not exist. If there is a Content-Length
     -- field, it is assumed that this size is correct.
     -- Line lengths should remain within the limit set by RFC
     -- 822, i.e. no more than 998 characters, and preferably no
     -- more than 78 (this length excludes the CRLF).
   append_urlencoded_to_string (*s*: *STRING*)
     -- Append fields of this MIME structure to *s*, and the body
     -- as x-www-form-urlencoded. Message must conform to RFC 2388.
feature(*s*) **from** *EPX_MIME_PART*
   -- Access
   as_string: *STRING*
     -- Serialized MIME message
   auto_insert_content_length: *BOOLEAN*
     -- If a Content-Length field does not exist, should
     -- *append_to_string* automatically add one?
   body: *EPX_MIME_BODY*
     -- The body, can be multipart
   header: *EPX_MIME_HEADER*
     -- Fields for this part
   multipart_body: *EPX_MIME_BODY_MULTIPART*
     -- *body* if *body* contains multiple parts, Void otherwise
   text_body: *EPX_MIME_BODY_TEXT*
     -- *body* if *body* is a text body and not multi-part, Void otherwise
feature(*s*) **from** *EPX_MIME_PART*
   -- Status
   is_valid: *BOOLEAN*
     -- Does this message conform to the MIME specification?
     -- If so, it can be serialized.
     -- If the body is multi-part, the boundary must be set.
feature(*s*) **from** *EPX_MIME_PART*
   -- Body creation/removal
   clear_body
     -- Set *body* to Void.
   create_multipart_body
     -- Set *body* to a container.
   create_singlepart_body
     -- Set *body* to a single part body.

-- If we find a Content-Disposition field with a filename
-- parameter, body data wil be saved to a temporary file when
-- set, insted of kept in memory.
feature(*s*) **from** *EPX__MIME__PART*
    -- Change
  set__auto__insert__content__length (*a__value*: *BOOLEAN*)
    -- Set if Content-Length fields should be automatically
    -- supplied, if onen doesnt exist, in *append__to__string*.
invariant
  header__not__void: *header /= Void*;
  bodies__in__sync: *body /= Void* **implies** *body.is__multipart = (multipart__body /= Void)*
and not *body.is__multipart = (text__body /= Void)*;
end *of EPX__MIME__PART*

E.4 Short form of EPX_SOAP_WRITER

class *interface EPX_SOAP_WRITER*
creation
   make
      -- Create an XML document with initial capacity of 1024 characters.
   make_with_capacity (*a_capacity*: *INTEGER*)
      -- Create an XML document with initial capacity of
      -- *a_capacity* characters.
feature(*s*) **from** *EPX_SOAP_WRITER*
   -- SOAP specific calls
   start_envelope
   stop_envelope
   start_header
   stop_header
   start_body
   stop_body
feature(*s*) **from** *EPX_SOAP_WRITER*
   -- SOAP header attributes
   set_must_understand (*value*: *BOOLEAN*)
      -- Set the SOAP-Env:mustUnderstand attribute to *value*.
feature(*s*) **from** *EPX_SOAP_WRITER*
   -- Queries if tags started
   is_envelope_started: *BOOLEAN*
   is_header_started: *BOOLEAN*
   is_body_started: *BOOLEAN*
feature(*s*) **from** *EPX_SOAP_WRITER*
   -- SOAP tags
   soap_env_body: *STRING*
   soap_env_envelope: *STRING*
   soap_env_header: *STRING*
feature(*s*) **from** *EPX_SOAP_WRITER*
   -- SOAP name space
   soap_env: *STRING*
   soap_name_space: *STRING*
invariant
   -- lower_a_code_definition: lower_a_code = (a).code
   -- Same thing for all other codes.
   -- (see "note" in indexing clause.)
   accessing_real_singleton: *security_is_real_singleton*;
   valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
   my_xml_not_void: *my_xml* /= *Void*;
   same_size: *attributes.count* = *values.count*;
   has_tag_stack: *tags* /= *Void*;
   comparing_references_is_not_good_enough: *tags.equality_tester* /= *Void*;
   fragment_has_no_header: *is_fragment* **implies** *is_header_written*;
   values_not_void: *values* /= *Void*;
   attributes_not_void: *attributes* /= *Void*;
   every_attribute_has_a_value: *attributes.count* = *values.count*;

tag_state_valid: *tag_state* $>=$ *tag_pending* **and** *tag_state* $<=$ *tag_closed*;
tag_started_and_pending_in_sync: *tag_state* $=$ *tag_pending* **implies** *is_tag_started*;
tag_closed_is_not_started: *tag_state* $=$ *tag_closed* $=$ **not** *is_tag_started*;
end *of EPX_SOAP_WRITER*

E.5 Short form of EPX_XML_WRITER

class *interface EPX_XML_WRITER*
creation
  make
    -- Create an XML document with initial capacity of 1024 characters.
  make_with_capacity (*a_capacity*: *INTEGER*)
    -- Create an XML document with initial capacity of
    -- *a_capacity* characters.
  make_fragment
    -- Create an XML fragment (document without header) with
    -- initial capacity of 1024 characters.
  make_fragment_with_capacity (*a_capacity*: *INTEGER*)
    -- Create an XML fragment (document without header) with
    -- initial capacity of *a_capacity* characters.
feature(*s*) **from** *EPX_XML_WRITER*
  -- Initialization
  make
    -- Create an XML document with initial capacity of 1024 characters.
  make_fragment
    -- Create an XML fragment (document without header) with
    -- initial capacity of 1024 characters.
  make_with_capacity (*a_capacity*: *INTEGER*)
    -- Create an XML document with initial capacity of
    -- *a_capacity* characters.
  make_fragment_with_capacity (*a_capacity*: *INTEGER*)
    -- Create an XML fragment (document without header) with
    -- initial capacity of *a_capacity* characters.
feature(*s*) **from** *EPX_XML_WRITER*
  -- Status
  can_add_attributes: *BOOLEAN*
    -- Can attributes be added to the current tag?
  is_a_parent (*tag*: *STRING*): *BOOLEAN*
    -- Is *tag* the current element, or is it a parent of the
    -- current tag at some point?
  is_attribute_set (*a_name_space, an_attribute*: *STRING*): *BOOLEAN*
    -- Has *an_attribute* been given a value?
  is_element_with_data: *BOOLEAN*
    -- Has data been added to this element or in case this
    -- element has not yet been written, has data been added to
    -- its parents element?
  is_fragment: *BOOLEAN*
    -- Is the XML document being created a fragment?
  is_header_written: *BOOLEAN*
    -- Is the XML header is written or is this a fragment that
    -- does not need a header?
  is_indenting: *BOOLEAN*
    -- When XML is written, is an attempt made to beautify the
    -- results? This is the default.

      -- Indented XML files are more readable, but it can create
      -- invalid XML, because the schema is not known. It also
      -- slows down writing the XML.
is_ns_started (*a_name_space, a_tag*: *STRING*): *BOOLEAN*
      -- Is name_space:tag the current element?
is_started (*a_tag*: *STRING*): *BOOLEAN*
      -- Is *tag* the current element?
is_tag_started: *BOOLEAN*
      -- Is there an unclosed element?
feature(*s*) **from** *EPX_XML_WRITER*
   -- Access
as_string: *STRING*
      -- The result as plain STRING
as_uc_string: *UC_STRING*
      -- The result as Unicode string, i.e. UC_STRING
tag_depth: *INTEGER*
      -- Number of tags that have been started, but not stopped
unfinished_xml: *STRING*
      -- The *xml* in progress
feature(*s*) **from** *EPX_XML_WRITER*
   -- Change
clear
      -- Start fresh.
      --   local
      --   s: STRING
set_indenting (*an_indenting*: *BOOLEAN*)
feature(*s*) **from** *EPX_XML_WRITER*
   -- Commands that expand *xml*
add_attribute (*an_attribute, a_value*: *STRING*)
      -- Add an attribute of the current tag. Attribute cannot be
      -- modified later unlike *set_attribute* and *a_value* does
      -- not have to be cloned if you want to reuse that STRING.
      -- *attribute* must be name-space less, else use *add_ns_attribute*.
      -- *value* may not contain an entity reference.
add_a_name_space (*a_prefix, a_uri*: *STRING*)
      -- Define a name space.
add_cdata (*a_data*: *STRING*)
      -- Add data within a CDATA tag. With CDATA much less
      -- meta-characters have to be quoted in case *a_data* is
      -- itself XML.
add_data (*a_data*: *STRING*)
      -- Write data in the current tag.
      -- Invalid characters like < or > are quoted.
      -- Use *add_raw* if you dont want quoting.
      -- This routine is not safe when writing data inside comments.
puts (*a_data*: *STRING*)
      -- Write data in the current tag.
      -- Invalid characters like < or > are quoted.
      -- Use *add_raw* if you dont want quoting.

   -- This routine is not safe when writing data inside comments.
add_entity (*an_entity_name*: *STRING*)
   -- Write entity *name* as element data.
add_header (*encoding*: *STRING*)
   -- Add the XML header, document is encoded in
   -- *encoding*. Making sure this encoding is followed, is the
   -- responsibility of the client.
add_header_iso_8859_1_encoding
   -- Document is iso-8859-1 encoded.
add_header_utf_8_encoding
   -- Document is utf8 encoded.
add_name_space (*a_prefix, a_uri*: *STRING*)
   -- Define a name space.
add_ns_attribute (*a_name_space, an_attribute, a_value*: *STRING*)
   -- Add an attribute to the current tag. This attribute cannot
   -- be changed later, use *set_ns_attribute* for that.
   -- *a_value* does not have to be cloned if you want to reuse
   -- that STRING.
   -- *value* may not contain an entity reference. *name_space*
   -- is the optional prefix to be used, not the actual URI.
add_raw (*raw_data*: *STRING*)
   -- Write *raw_data* straight in the current tag, meta
   -- characters are not quoted, control characters are not
   -- checked, etc.
add_system_doctype (*root_tag, system_id*: *STRING*)
   -- Add a <!DOCTYPE element.
   -- Only allowed when no tags have been written.
add_tag (*tag, a_data*: *STRING*)
   -- Shortcut for *add_tag*, *add_data* and *stop_tag*.
add_ns_tag (*name_space, tag, a_data*: *STRING*)
   -- Shortcut for *add_ns_tag*, *add_data* and *stop_tag*.
get_attribute (*attribute*: *STRING*): *STRING*
   -- Get contents of attribute *attribute* for
   -- current tag. *attribute* may include a name space.
   -- Returns Void if attribute doesnt exist
put (*a*: *ANY*)
   -- Write data within the current tag.
put_new_line
   -- Add a new line in the current tag.
set_attribute (*attribute, a_value*: *STRING*)
   -- Set an attribute of the current tag.
   -- *attribute* must be name-space less, else use *set_ns_attribute*.
   -- *value* may not contain an entity reference.
   -- As the attribute is not immediately written, make sure
   -- *attribute* and *value* do not change (ie are cloned or
   -- immutable).
set_a_name_space (*a_prefix, a_uri*: *STRING*)
   -- Define a name space.
   -- As the attribute is not immediately written, make sure

        -- *a__prefix* and *a__uri* do not change (ie are cloned or
        -- immutable).
set__default__name__space (*uri*: *STRING*)
        -- Set the default name space.
set__ns__attribute (*name__space, attribute, value*: *STRING*)
        -- Set an attribute of the current tag. *value* may not
        -- contain an entity reference. *name__space* is the optional
        -- prefix to be used, not the actual URI.
        -- As the attribute is not immediately written, make sure
        -- *name__space*, *attribute* and *value* do not change (ie
        -- are cloned or immutable).
start__ns__tag (*name__space, tag*: *STRING*)
        -- Start a new tag in the given *name__space. name__space* is
        -- a prefix only, not the actual URI. If *name__space* is Void
        -- or empty, the tag will not get a prefix.
        -- As the tag is not immediately written, be sure that *tag*
        -- does not change (ie is cloned or immutable) if
        -- *name__space* is Void or empty.
start__tag (*tag*: *STRING*)
        -- Start a new tag.
        -- As the tag is not immediately written, make sure *tag*
        -- does not change (ie is cloned or immutable).
stop__tag
        -- Stop last started tag.
feature(*s*) **from** *EPX__XML__WRITER*
    -- Quote unsafe characters
replace__content__meta__characters (*s*: *STRING*)
        -- Replace all characters in *s* that have a special meaning in
        -- XML. These characters are < and & and the sequence "]]>".
        -- This routine is slow when *data* is actually a UC__STRING
        -- and is very large. Moving bytes to the right to insert the
        -- quoting characters takes up a very long time.
feature(*s*) **from** *EPX__XML__WRITER*
    -- Comments
add__comment (*a__comment*: *STRING*)
        -- Add a comment.
        -- This routine does not yet quote meta data properly. Need a
        -- separate comment state to properly quote meta data inside
        -- comments.
start__comment
        -- Write the XML comment start tag.
stop__comment
        -- Stop a started XML comment.
invariant
    -- lower__a__code__definition: lower_a_code = (a).code
    -- Same thing for all other codes.
    -- (see "note" in indexing clause.)
accessing__real__singleton: *security__is__real__singleton*;
valid__error__action: *error__action >= 0* **and** *error__action <= 2;*

    my_xml_not_void: *my_xml* /= *Void*;
    same_size: *attributes.count* = *values.count*;
    has_tag_stack: *tags* /= *Void*;
    comparing_references_is_not_good_enough: *tags.equality_tester* /= *Void*;
    fragment_has_no_header: *is_fragment* **implies** *is_header_written*;
    values_not_void: *values* /= *Void*;
    attributes_not_void: *attributes* /= *Void*;
    every_attribute_has_a_value: *attributes.count* = *values.count*;
    tag_state_valid: *tag_state* >= *tag_pending* **and** *tag_state* <= *tag_closed*;
    tag_started_and_pending_in_sync: *tag_state* = *tag_pending* **implies** *is_tag_started*;
    tag_closed_is_not_started: *tag_state* = *tag_closed* = **not** *is_tag_started*;
  end *of EPX_XML_WRITER*

E.6 Short form of EPX_XHTML_WRITER

class *interface EPX_XHTML_WRITER*
creation
  make
    -- Create an XML document with initial capacity of 1024 characters.
  make_with_capacity (*a_capacity*: *INTEGER*)
    -- Create an XML document with initial capacity of
    -- *a_capacity* characters.
  make_fragment
    -- Create an XML fragment (document without header) with
    -- initial capacity of 1024 characters.
  make_fragment_with_capacity (*a_capacity*: *INTEGER*)
    -- Create an XML fragment (document without header) with
    -- initial capacity of *a_capacity* characters.
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- overrule some xml stuff
  new_line_after_closing_tag (*a_tag*: *STRING*)
    -- Outputs a new line, called when *a_tag* is closed
    -- can be overridden to start a new line only occasionally
    -- For XHTML documents a new line is treated as a single
    -- space, so it can influence layout.
  new_line_before_starting_tag (*a_tag*: *STRING*)
    -- Outputs a new line, called when *a_tag* is about to begin.
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- doctype
  doctype
    -- Default doctype is *doctype_strict*.
  doctype_frameset
    -- Output will be frame-based.
  doctype_strict
    -- Output will be strict XHTML in the ISO-8859-1 encoding.
  doctype_strict_utf8
    -- Output will be strict XHTML in the UTF-8 encoding.
  doctype_transitional
    -- Output will be transitional XHTML with ISO-8859-1 encoding.
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Set well-known attributes
  set_id (*a_id*: *STRING*)
    -- Set the id attribute.
  set_xhtml_name_space
    -- Add the XHTML name space to the current tag.
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Page
  b_html
  e_html
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Header
  meta_content_type (*a_content_type*: *STRING*)

    -- Add Content-Type to HTML. *a_content_type* is of the
    -- format "text/html; charset=utf-8".
  meta_refresh_other (*a_time*: *INTEGER*; *a_url*: *STRING*)
  b_head
  e_head
  title (*a_text*: *STRING*)
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Body
  b_body
  e_body
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Section headers
  h1 (*header_text*: *STRING*)
  h2 (*header_text*: *STRING*)
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Paragraph
  br
    -- break.
  br_clear_all
    -- Add break and flush all floats.
  b_p
  e_p
  p (*par*: *STRING*)
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Inline tags
  b_b
    -- Begin bold font.
  e_b
    -- End bold font.
  b_i
    -- Begin italic font.
  e_i
    -- End italic font.
  b_tt
    -- teletype writer font
  e_tt
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Lists
  b_ul
    -- Begin unordered list.
  e_ul
    -- End unordered list.
  b_li
    -- Begin list item.
  e_li
    -- End list item.
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Quotes
  b_blockquote

e_blockquote

blockquote (*a_quote*: *STRING*)

feature(*s*) **from** *EPX_XHTML_WRITER*

  -- Link

b_a (*href*: *STRING*)

e_a

a (*href, s*: *STRING*)

feature(*s*) **from** *EPX_XHTML_WRITER*

  -- Rules

hr

    -- horizontal rule

feature(*s*) **from** *EPX_XHTML_WRITER*

  -- White space

nbsp

    -- Add a non breaking white space.

feature(*s*) **from** *EPX_XHTML_WRITER*

  -- Verbatim

b_pre

e_pre

feature(*s*) **from** *EPX_XHTML_WRITER*

  -- Images

b_img (*a_src, a_description*: *STRING*)

    -- Start an img tag with *a_src* the source of the image and

    -- *a_description* the alternative (alt) text of the image.

e_img

    -- Stop image.

img (*a_src, a_description*: *STRING*)

    -- Emit an img tag with *a_src* the source of the image and

    -- *a_description* the alternative (alt) text of the image.

feature(*s*) **from** *EPX_XHTML_WRITER*

  -- Tables

b_table

    -- Begin a table.

e_table

    -- End a table.

b_tr

    -- Begin a row.

e_tr

    -- End a row.

td (*a_content*: *STRING*)

    -- Add cell with optional contents.

b_td

    -- Begin a column.

e_td

    -- End a column.

th (*a_title*: *STRING*)

    -- Add a header cell.

b_th

    -- Begin a table header cell.

e_th
  -- Add a table header cell.
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Forms
b_form (*method, action*: *STRING*)
b_form_get (*action*: *STRING*)
b_form_post (*action*: *STRING*)
e_form
b_input (*type, name*: *STRING*)
e_input
hidden (*name, value*: *STRING*)
b_button_submit (*name, value*: *STRING*)
e_button_submit
button_submit (*name, value*: *STRING*)
  -- Submit button.
b_button_reset
e_button_reset
button_reset
b_checkbox (*name, value*: *STRING*)
e_checkbox
label (*a_label, a_for*: *STRING*)
  -- Emit label tag *a_label* for a control with id *a_for*.
b_radio (*name, value*: *STRING*)
e_radio
b_select (*name*: *STRING*)
e_select
b_option
e_option
option (*text*: *STRING*)
selected_option (*choice*: *STRING*)
b_textarea (*name*: *STRING*)
  -- Begin multiline input control.
e_textarea
  -- End multiline input control.
input_text (*name*: *STRING*; *size*: *INTEGER*; *value*: *STRING*)
  -- Single line input.
b_input_text (*name*: *STRING*; *size*: *INTEGER*; *value*: *STRING*)
  -- Single line input.
e_input_text
  -- End single line input.
input_password (*name*: *STRING*; *size*: *INTEGER*; *value*: *STRING*)
  -- Single line password input.
feature(*s*) **from** *EPX_XHTML_WRITER*
  -- CSS style sheet support
b_style
  -- Start inline style.
e_style
set_class (*name*: *STRING*)
  -- set attribute class

set_style (*an_inline_style*: *STRING*)
    -- Set the style attribute.
style_sheet (*a_location, a_description, a_media*: *STRING*)
    -- Put in a link to refer to an external style sheet on disk.
    -- *a_media* is the intended destination medium for style
    -- information. It may be a single media descriptor or a
    -- comma-separated list. The default value for this attribute
    -- is "screen".
alternate_style_sheet (*a_location, a_description, a_media*: *STRING*)
    -- Put in a link to refer to an alternative style sheet.
    -- *a_media* is the intended destination medium for style
    -- information. It may be a single media descriptor or a
    -- comma-separated list. The default value for this attribute
    -- is "screen".

feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Link
link (*a_href, a_forward_link_types, a_backward_link_types, a_content_type, a_title, a_media*: *STRING*)
    -- Add a <link> element. This is used for document relationships.

feature(*s*) **from** *EPX_XHTML_WRITER*
  -- Divisions
b_div
    -- Start a <div> tag.
e_div
    -- Stop the <div> tag.
b_span
    -- Start a <span> tag.
e_span
    -- Stop the <span> tag.

feature(*s*) **from** *EPX_XHTML_WRITER*
  -- JavaScript support
b_external_script (*a_src*: *STRING*; *a_defer_execution*: *BOOLEAN*)
    -- Include external script. If *a_defer_execution* then
    -- browser may defer execution of script until page is
    -- rendered. This can improve performance.
b_script
    -- Start JavaScript.
e_script
external_script (*a_src*: *STRING*; *a_defer_execution*: *BOOLEAN*)
    -- Include external script. If *a_defer_execution* then
    -- browser may defer execution of script until page is
    -- rendered. This can improve performance.
set_onclick (*an_action*: *STRING*)

feature(*s*) **from** *EPX_XHTML_WRITER*
  -- HTML tag names
once_a: *STRING*
once_blockquote: *STRING*
once_body: *STRING*
once_br: *STRING*

once_div: *STRING*
once_form: *STRING*
once_h1: *STRING*
once_h2: *STRING*
once_h3: *STRING*
once_head: *STRING*
once_html: *STRING*
once_img: *STRING*
once_input: *STRING*
once_label: *STRING*
once_link: *STRING*
once_meta: *STRING*
once_option: *STRING*
once_p: *STRING*
once_pre: *STRING*
once_script: *STRING*
once_select: *STRING*
once_span: *STRING*
once_table: *STRING*
once_td: *STRING*
once_textarea: *STRING*
once_tr: *STRING*
once_title: *STRING*

feature(*s*) **from** *EPX_XHTML_WRITER*
-- Attribute values
once_selected: *STRING*
once_submit: *STRING*
once_text: *STRING*

invariant
-- lower_a_code_definition: lower_a_code = (a).code
-- Same thing for all other codes.
-- (see "note" in indexing clause.)
accessing_real_singleton: *security_is_real_singleton*;
valid_error_action: *error_action* $>=$ *0* **and** *error_action* $<=$ *2*;
my_xml_not_void: *my_xml* $/=$ *Void*;
same_size: *attributes.count* $=$ *values.count*;
has_tag_stack: *tags* $/=$ *Void*;
comparing_references_is_not_good_enough: *tags.equality_tester* $/=$ *Void*;
fragment_has_no_header: *is_fragment* **implies** *is_header_written*;
values_not_void: *values* $/=$ *Void*;
attributes_not_void: *attributes* $/=$ *Void*;
every_attribute_has_a_value: *attributes.count* $=$ *values.count*;
tag_state_valid: *tag_state* $>=$ *tag_pending* **and** *tag_state* $<=$ *tag_closed*;
tag_started_and_pending_in_sync: *tag_state* $=$ *tag_pending* **implies** *is_tag_started*;
tag_closed_is_not_started: *tag_state* $=$ *tag_closed* $=$ **not** *is_tag_started*;

end *of EPX_XHTML_WRITER*

Classes in this appendix build upon the abstract layer and generally need network access.

F.1 Short form of EPX_HOST_PORT

class *interface EPX_HOST_PORT*
creation
    make (*a_host*: *EPX_HOST*; *a_service*: *EPX_SERVICE*)
        -- Initialize socket for resolved host, using its first ip
        -- address.
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
    -- The singleton, available to any because its used in preconditions
    security: *STDC_SECURITY*
        -- Singleton entry point for security.
feature(*s*) **from** *STDC_BASE*
    -- Access
    errno: *STDC_ERRNO*
        -- Access to the variable that contains the error that occurred.
feature(*s*) **from** *STDC_BASE*
    -- Status
    raise_exception_on_error: *BOOLEAN*
        -- Should an exception be raised when an error occurs?
        -- If not, you have to check errno for any errors.
feature(*s*) **from** *STDC_BASE*
    -- Change
    set_default_action_on_error
        -- Use *security.error_handling.exceptions_enabled* to
        -- determine if an exception should be raised when a C call
        -- returns an error.
    set_raise_exception_on_error
        -- Always raise an exception when a C call returns an error.
    set_continue_on_error
        -- Never raise an exception when a C call returns an error.
    inherit_error_handling (*an_instance*: *STDC_BASE*)

-- Handle errors like *an_instance*
feature(*s*) **from** *EPX_HOST_PORT*
  -- Access
  host: *EPX_HOST*
    -- Resolved host name.
  service: *EPX_SERVICE*
    -- Port and protocol (udp/tcp) type.
  socket_address: *ABSTRACT_SOCKET_ADDRESS_IN_BASE*
    -- The socket address struct to be used by *connect*.
feature(*s*) **from** *EPX_HOST_PORT*
  -- Fill socket structure, so *ptr* returns something valid
  set_address (*item*: *INTEGER*)
    -- Use the ip address at *item* of *host* as the socket
    -- address.
invariant
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
  host_resolved: *host* /= *Void* **and then** *host.found*;
  has_service: *service* /= *Void*;
  socket_address_not_void: *socket_address* /= *Void*;
  address_type_matches: *host.address_family* = *socket_address.address_family*;
  port_matches: *service.port* = *socket_address.port*;
end *of EPX_HOST_PORT*

F.2 Short form of EPX_HTTP_10_CLIENT

class *interface EPX_HTTP_10_CLIENT*
creation
   make (*a_server_name*: *STRING*)
     -- Prepare for request to *a_server_name.*
   make_from_port (*a_server_name*: *STRING*; *a_port*: *INTEGER*)
   make_with_port (*a_server_name*: *STRING*; *a_port*: *INTEGER*)
     -- Prepare for request.
     -- Use *port* is 0 to use the default port (80).
   make_from_host (*a_host*: *EPX_HOST*)
     -- Prepare for request to resolved *a_host.* If *port* is 0,
     -- the default port is taken, else the port can be overruled.
   make_from_host_and_port (*a_host*: *EPX_HOST*; *a_port*: *INTEGER*)
     -- Prepare for request to *a_host.* If *port* is 0, the
     -- default port is taken, else the port can be overruled.
   make_secure (*a_server_name*: *STRING*)
     -- Prepare for secure (SSL) request to *a_host.*
   make_secure_with_port (*a_server_name*: *STRING*; *a_port*: *INTEGER*)
     -- Prepare for secure (SSL) request to *a_host.*
  feature(*s*) **from** *EPX_HTTP_10_CLIENT*
   -- Access
   client_version: *STRING*
     -- Client http version
   last_data: *EPX_MIME_PART*
     -- Data of last request send to server;
     -- Used by *read_response_with_redirect* to properly redirect
     -- a request.
   last_verb: *STRING*
     -- Verb of last request send to server;
     -- Used by *read_response_with_redirect* to properly redirect
     -- a request.
  feature(*s*) **from** *EPX_HTTP_10_CLIENT*
   -- Status
   reuse_connection: *BOOLEAN*
     -- Should HTTP connection be reused for more than 1 request?
  feature(*s*) **from** *EPX_HTTP_10_CLIENT*
   -- Requests
   delete (*a_request_uri*: *STRING*; *a_delete_data*: *EPX_MIME_PART*)
     -- Send DELETE request to server.
     -- Use *read_response* to fetch the response and actual response code.
   get (*a_request_uri*: *STRING*)
     -- Send GET request to server.
     -- Sets *response_code* to 200 if the request was send successfully.
     -- If sending the request failed (usually because the server refused
     -- the connection), 503 is returned.
     -- Use *read_response* to fetch the response and actual response code.
   head (*a_request_uri*: *STRING*)
     -- Send HEAD request to server.

       -- *a_request_uri* should not include http: and the host name, only
       -- the page that is requested. Any query and fragment parts are ok.
       -- Sets *response_code* to 200 if the request was send successfully.
       -- If sending the request failed (usually because the server refused
       -- the connection), 503 is returned.
       -- Use *read_response* to fetch the response and actual response code.
  options (*a_request_uri*: *STRING*)
       -- Get server options. *a_request_uri* is required when the
       -- request is being made to a proxy.
       -- Sets *response_code* to 200 if the request was send successfully.
       -- If sending the request failed (usually because the server refused
       -- the connection), 503 is returned.
       -- Use *read_response* to fetch the response and actual response code.
  put (*a_request_uri*: *STRING*; *a_put_data*: *EPX_MIME_PART*)
       -- Put *a_put_data* to *host* using the HTTP PUT request.
       -- Sets *response_code* to 200 if the request was send successfully.
       -- If sending the request failed (usually because the server refused
       -- the connection), 503 is returned.
       -- Use *read_response* to fetch the response and actual response code.
       -- Tip: use EPX_MIME_FORM.make_form_data to build the
       -- most common form data messages.
  post (*a_request_uri*: *STRING*; *a_post_data*: *EPX_MIME_PART*)
       -- Post *a_post_data* to *host* using the HTTP POST
       -- request. *a_post_data* may be empty in which case no data
       -- will be send with this POST request.
       -- Sets *response_code* to 200 if the request was send successfully.
       -- If sending the request failed (usually because the server refused
       -- the connection), 503 is returned.
       -- Use *read_response* to fetch the response and actual response code.
       -- Tip: use EPX_MIME_FORM.make_form_data to build the
       -- most common form data messages.
       -- Tip 2: *post_data_content_type_recognized* is usually true if
       -- you sent data to an HTTP server.
  post_xml (*a_request_uri*: *STRING*; *a_post_data*: *STRING*)
       -- Post *a_post_data* to *host* using the HTTP POST request.
       -- Sets *response_code* to 200 if the request was send successfully.
       -- If sending the request failed (usually because the server refused
       -- the connection), 503 is returned.
       -- Use *read_response* to fetch the response and actual response code.
       -- *a_post_data* should be valid XML.
  send_request (*a_verb*, *a_request_uri*: *STRING*; *a_request_data*: *EPX_MIME_PART*)
       -- Send request *a_verb* with *a_request_uri* to *host*.
       -- Additional header fields and an optional body can be passed in
       -- *a_request_data*.
       -- Sets *response_code* to 200 if the request was send successfully.
       -- If sending the request failed (usually because the server refused
       -- the connection), 503 is returned.
feature(*s*) **from** *EPX_HTTP_10_CLIENT*
  -- Authentication response

is__authentication__required: *BOOLEAN*
    -- Set if response from server indicates that proper
    -- authentication is required
authentication__realm: *STRING*
    -- Realm of authentication if defined; but according to the
    -- spec all schemes should have one.
authentication__scheme: *STRING*
    -- Required authentication scheme
feature(*s*) **from** *EPX__HTTP__10__CLIENT*
  -- Authentication setup
basic__authentication: *STRING*
    -- Optional authentication header to send with a request
set__basic__authentication (*a__user__name, a__password*: *STRING*)
    -- Make sure the Authorization header is included in the
    -- request.
feature(*s*) **from** *EPX__HTTP__10__CLIENT*
  -- Cookies
cookies: *DS__HASH__TABLE*[*EPX__HTTP__COOKIE,STRING*]
    -- Cookies that will be sent with the request to the server
set__cookie (*a__cookie__name, a__cookie__value*: *STRING*)
    -- Add or update a cookie that will be send to the browser
    -- then *context__text__html* is called.
wipe__out__cookies
    -- Remove all cookies
feature(*s*) **from** *EPX__HTTP__10__CLIENT*
  -- Fields that are send with a request if set
accept: *STRING*
    -- What kind of output can the client handle?
    -- Examples are:
    --   Accept: text/plain; q=0.5, text/html,
    --   text/x-dvi; q=0.8, text/x-c
user__agent: *STRING*
    -- Identification of client program;
    -- Common examples are:
    --   Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
    --   Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020529
    --   Microsoft Internet Explorer
set__accept (*value*: *STRING*)
    -- Set the media types which are acceptable for the response.
set__user__agent (*value*: *STRING*)
    -- Set the client identification.
feature(*s*) **from** *EPX__HTTP__10__CLIENT*
  -- Response
body: *EPX__MIME__BODY__TEXT*
    -- Body as text, if applicable, else Void
fields: *DS__HASH__TABLE*[*EPX__MIME__FIELD,STRING*]
    -- Header fields of response
is__response__ok: *BOOLEAN*
    -- Does the returned *response__code* indicate success?

last_uri: *STRING*
    -- URI of last request
read_response
    -- Read entire response and make it available in
    -- *response*. Header is available in *fields*, and text body, if
    -- any in *body*.
    -- If a redirect response is returned, the redirect is not
    -- automatically read. Use *read_response_with_redirect* to
    -- automatically handle redirects.
    -- If the server has returned an invalid response, the
    -- *response_code* is set to 500.
read_response_header
    -- Read the header and make it available in
    -- *response*. Header is available in *fields*. Due to
    -- buffering first part of body is usually also available in
    -- *response.body*.
    -- If a redirect response is returned, the redirect is not
    -- automatically read. Use *read_response_with_redirect* to
    -- automatically handle redirects.
    -- If the server has returned an invalid response, the
    -- *response_code* is set to 500.
    -- First part of body is made available in *first_body_part*
read_response_with_redirect
    -- As *read_response*, but if a redirect response code is
    -- received, request is automatically redirected.
    -- It assumes *last_verb* contains the verb of the last
    -- request send.
    -- A maximum of twenty redirects are followed, after that
    -- this routine just returns.
response: *EPX_MIME_PART*
    -- The entire parsed MIME message;
    -- It is set by *read_response*. May be Void if there is no body.
response_phrase: *STRING*
    -- HTTP server response phrase;
    -- set by *read_response*.
server_version: *STRING*
    -- HTTP server version;
    -- set by *read_response*.
feature(*s*) **from** *EPX_HTTP_10_CLIENT*
  -- Individual response fields, Void if not available
location: *STRING*
    -- The contents of the Location field in the header, if any
invariant
  three_digit_reply_code: *response_code = 0* **or else** *response_code >= 100* **and** *response_code*
<= 999;
  accessing_real_singleton: *security_is_real_singleton*;
  valid_error_action: *error_action >= 0* **and** *error_action <= 2*;
  service_not_void: *http_service /= Void*;
  socket_void_or_connected: *http = Void* **or else** *http.is_open*;

connected__is__readable: *http* /= *Void* **implies** *http.is__open__read*;
open__implies__resolved: *is__open* **implies** *is__resolved*;
valid__server__name: *server__name* /= *Void* **and then not** *server__name.is__empty*;
is__valid__user: *is__valid__user__name*(*user__name*);
is__valid__password: *is__valid__password*(*password*);
have__www__authenticate__header__if__authentication__required: *is__authentication__required*
implies *response.header.has*(*field__name__www__authenticate*);
end *of EPX__HTTP__10__CLIENT*

F.3 Short form of EPX_IMAP4_CLIENT

class *interface EPX_IMAP4_CLIENT*
creation
  make (*a_host*: *STRING*)
    -- Initialize client and try to open connection to imap server.
    -- Check *is_open* if could connect to server.
    -- If not, *a_host* might not be resolvable.
  make_with_port (*a_host*: *STRING*; *a_port*: *INTEGER*; *a_secure*: *BOOLEAN*)
    -- Initialize client and try to open connection to imap
    -- server at *a_host*.
    -- If *a_port* is zero, use the default port for an insecure
    -- or secure connection, depending on *a_secure*.
    -- Check *is_open* if could connect to server. If not,
    -- *a_host* might not be resolvable.
  make_secure (*a_host*: *STRING*)
    -- Initialize client and try to open connection to imap server.
    -- Check *is_open* if could connect to server.
    -- If not, *a_host* might not be resolvable.
feature(*s*) **from** *EPX_IMAP4_CLIENT*
  -- Open/close
  open
    -- Open connection to an imap server. On success *is_open* is
    -- True. If there is a failure, check *error_message* for any
    -- human readable error message.
  close
    -- Close connection to imap server.
feature(*s*) **from** *EPX_IMAP4_CLIENT*
  -- Access
  error_message: *STRING*
    -- Human readable error message when *open* fails; warning:
    -- might be Void even when there is an error!
  host_name: *STRING*
    -- Name of server running the imap daemon
  port: *INTEGER*
    -- Port at *host_name*
  response: *EPX_IMAP4_RESPONSE*
    -- Responses received by server.
  state: *EPX_IMAP4_STATE*
    -- Current state, one of four
feature(*s*) **from** *EPX_IMAP4_CLIENT*
  -- Status
  is_open: *BOOLEAN*
    -- Is client connected to IMAP server?
  is_secure_connection: *BOOLEAN*
    -- Do we have a secure connection to server?
feature(*s*) **from** *EPX_IMAP4_CLIENT*
  -- Not-authenticated state commands
  login (*a_user_name, a_password*: *STRING*)

        -- Login to the IMAP server using _a__user__name_ and
        -- _a__password_. If login successful, then _state_ will be
        -- set to _Authenticated__state_. If login was unsuccessful,
        -- see _login__failure__reason_ for a human readable error message.
     noop
        -- Since any command can return a status update as untagged
        -- data, the NOOP command can be used as a periodic poll for
        -- new messages or message status updates during a period of
        -- inactivity. The NOOP command can also be used to reset
        -- any inactivity autologout timer on the server.
        -- A _noop_ can be issued in any state.
  feature(_s_) **from** _EPX__IMAP4__CLIENT_
    -- Authenticated state commands
    _create__mailbox_ (_a__mailbox__name_: _STRING_)
        -- The CREATE command creates a mailbox with the given name.
        -- An OK response is returned only if a new mailbox with that
        -- name has been created. It is an error to attempt to
        -- create INBOX or a mailbox with a name that refers to an
        -- extant mailbox.
    _delete__mailbox_ (_a__mailbox__name_: _STRING_)
        -- The DELETE command permanently removes the mailbox with
        -- the given name.
    examine (_a__mailbox__name_: _STRING_)
        -- The EXAMINE command is identical to SELECT and returns the
        -- same output; however, the selected mailbox is identified
        -- as read-only. No changes to the permanent state of the
        -- mailbox, including per-user state, are permitted.
    _get__delimiter_
        -- Make sure _response.delimiter_ has the correct value.
    _list__all_
        -- _list__all_ returns the complete set of all names available
        -- to the client.
    _list__subscribed_
        -- _list__subscribed_ returns the complete set of names that
        -- the user has declared as being "active" or "subscribed".
    _select__mailbox_ (_a__mailbox__name_: _STRING_)
        -- The SELECT command selects a mailbox so that messages in
        -- the mailbox can be accessed.
        -- If _response.is__ok_ then _response.current__mailbox_
        -- contains some information about the selected mailbox.
  feature(_s_) **from** _EPX__IMAP4__CLIENT_
    -- Selected state commands
    _check__mailbox_
        -- The CHECK command requests a checkpoint of the currently
        -- selected mailbox. A checkpoint refers to any
        -- implementation-dependent housekeeping associated with the
        -- mailbox (e.g. resolving the servers in-memory state of
        -- the mailbox with the state on its disk) that is not
        -- normally executed as part of each command. A checkpoint

    -- MAY take a non-instantaneous amount of real time to
    -- complete. If a server implementation has no such
    -- housekeeping considerations, CHECK is equivalent to NOOP.
    -- There is no guarantee that an EXISTS untagged response
    -- will happen as a result of CHECK. NOOP, not CHECK, SHOULD
    -- be used for new mail polling.
close_mailbox
    -- This command permanently removes from the currently
    -- selected mailbox all messages that have the \Deleted flag
    -- set, and returns to authenticated state from selected
    -- state.
copy_message (*sequence_number*: *INTEGER*; *to_mailbox_name*: *STRING*)
    -- Copy message with sequence_number *sequence_number* to the
    -- mailbox *to_mailbox_name*.
delete_message (*sequence_number*: *INTEGER*)
    -- Delete message with sequence_number *sequence_number* from
    -- the current mailbox.
expunge
    -- The EXPUNGE command permanently removes all messages that
    -- have the \Deleted flag set from the currently selected
    -- mailbox.
fetch (*a_set*: *STRING*; *a_format*: *STRING*)
    -- Fetch messages described by *a_set* in format described by
    -- *a_format*. Data is stored into a new
    -- *response.current_message* object.
fetch_body (*sequence_number*: *INTEGER*)
    -- Fetch message body, return raw RFC822 body in
    -- *last_body*.
fetch_header (*sequence_number*: *INTEGER*)
    -- Fetch just the message header (no flags for example),
    -- return raw RFC822 header in
    -- *response.current_message.header*.
fetch_header_and_flags (*sequence_number*: *INTEGER*)
    -- Fetch the message header and flags.
    -- Raw RFC822 header is in
    -- *response.current_message.header*; flags are in
    -- *response.current_message.flags*.
fetch_message (*sequence_number*: *INTEGER*)
    -- Fetch message, return raw RFC822 message in *response.message*.
fetch_size (*sequence_number*: *INTEGER*)
    -- Fetch message, return raw RFC822 size in *response.message_size*.
logout
    -- Inform the server that the client is done with the
    -- connection.
mark_unseen (*sequence_number*: *INTEGER*)
    -- Remove the \Seen flag from the given message.
    -- It does not update *current_message.flags* as it runs
    -- silently.
feature(*s*) **from** *EPX_IMAP4_CLIENT*

-- Selected state queries
is_valid_sequence_number (*a_number*: *INTEGER*): *BOOLEAN*
    -- Is *a_number* a valid sequence number for *current_mailbox*?
is_valid_mailbox_name (*a_name*: *STRING*): *BOOLEAN*
    -- Is *a_mailbox_name* a valid mailbox name?
    -- It should not be empty, and it should not have the double
    -- quote character in its name.
invariant
    accessing_real_singleton: *security_is_real_singleton*;
    valid_error_action: *error_action* >= *0* **and** *error_action* <= *2*;
    host_name_not_empty: *host_name* /= *Void* **and then not** *host_name.is_empty*;
    state_not_void: *state* /= *Void*;
    closed_implies_unauthenticated: **not** *is_open* **implies** *state.is_not_authenticated*;
    authenticated_implies_open: **not** *state.is_not_authenticated* **implies** *is_open*;
    response_not_void: *response* /= *Void*;
    selected_state_has_current_mailbox: *state.is_selected* **implies** *response.current_mailbox*
/= *Void*;
    unselected_state_has_no_current_mailbox: **not** *state.is_selected* **implies** *response.current_mailbox*
= *Void*;
    end *of EPX_IMAP4_CLIENT*

F.4 Short form of ULM_LOGGING

This class depends on Standard C only. It is the EPX_LOG_HANDLER that is platform specific. e-POSIX provides implementations of this class for Unix through syslog and for Windows through the NT event log.

class *interface ULM_LOGGING*
"2007-12-24: Please use the new NET_LOGGER classes"
end *of ULM_LOGGING*