POSIX Extension

version 0.90

Erick Gallesio
Université de Nice - Sophia Antipolis
930 route des Colles, BP 145
F-06903 Sophia Antipolis, Cedex
France

This document was produced using the Skribe Programming Language and its ConTEXt engine.
For further information on Skribe, see http://www-sop.inria.fr/mimosa/fp/Skribe/.

Document created on June 12, 2007.

1 Introduction

This extension exposes POSIX functionalities in STklos. Note that this version is **very incomplete** and only a few POSIX functionalities are available for now.

2 Basic Usage

To use this extension you need to include the following form in your program:

```
(require "stklos-posix")
(import stklos-posix)
```

The functions of this extension may set a special parameter value when an error occurs. The mechanism used is described here (see also `??` for more information). In general a function returns a useful result (or `#t` if no useful value is possible) if no error is detected. When a function detects an error, it returns `#f`. The error number is then available through the **(posix-errno)** parameter and a string describing the error can be built by the **(posix-error)** primitive.

3 Posix API

3.1 Directories Functions

`(posix-change-directory path)` STklos *procedure*

Change the current directory to "path". Return `#t` in case of sucess.

`(posix-current-directory)` STklos *procedure*

Return the current directory if possible. When an error occurs, this function returns `#f`.

`(posix-make-directory path)` STklos *procedure*

Create a directory with name "path". Return `#t` in case of sucess.

`(posix-delete-directory path)` STklos *procedure*

Delete the directory "path". Return `#t` in case of sucess.

3.2 Links and Symbolic Links functions

`(posix-make-link old-path new-path)` STklos *procedure*

Create a hard link with the filename "new-path" that points to the file named "old-path". Return `#t` in case of sucess.

```
(posix-delete-link old-path new-path)
```

STklos *procedure*

Create a symbolic link with the filename "new-path" that points to the file named "old-path". Return `#t` in case of sucess.

```
(posix-file-is-symbolic-link?  path)
```

STklos *procedure*

Return `#t` if "path" designates a symbolic link and `#f` othewise.

```
(posix-read-symbolic-link path)
```

STklos *procedure*

Return a string containing the filename to which the symbolic link "path" points to.

3.3 Error management

When an error occurs POSIX sets the `errno` global variable to a return code describing the error. This error code is available from the `posix-errno` Scheme parameter.

```
(posix-errno)
(posix-errno value)
```

STklos *procedure*

Return the value of the POSIX `errno` variable. When a value is passed to the `posix-errno`, this value is set to the POSIX `errno` variable.

```
(posix-error)
```

STklos *procedure*

Return a string describing the last POSIX error detected.

Furthermore, the POSIX extension defines also the following constants for representing POSIX error numbers.

posix/e2big	posix/eacces	posix/eaddrinuse
posix/eaddrnotavail	posix/eafnosupport	posix/eagain
posix/ealready	posix/ebadf	posix/ebadmsg
posix/ebusy	posix/ecanceled	posix/echild
posix/econnaborted	posix/econnrefused	posix/econnreset
posix/edeadlk	posix/edestaddrreq	posix/edom
posix/edquot	posix/eexist	posix/efault
posix/efbig	posix/ehostunreach	posix/eidrm
posix/eilseq	posix/einprogress	posix/eintr
posix/einval	posix/eio	posix/eisconn
posix/eisdir	posix/eloop	posix/emfile
posix/emlink	posix/emsgsize	posix/emultihop
posix/enametoolong	posix/enetdlown	posix/enetreset
posix/enetunreach	posix/enfile	posix/enobufs
posix/enodata	posix/enodev	posix/enoent
posix/enoexec	posix/enolck	posix/enolink

posix/enomem	posix/enomsg	posix/enoprotoopt
posix/enospc	posix/enosr	posix/enostr
posix/enosys	posix/enotconn	posix/enotdlir
posix/enotempty	posix/enotsock	posix/enotsup
posix/enotty	posix/enxio	posix/eopnotsupp
posix/eoverflow	posix/eperm	posix/epipe
posix/eproto	posix/eprotonosupport	posix/eprototype
posix/erange	posix/erofs	posix/espipe
posix/esrch	posix/estale	posix/etime
posix/etimedout	posix/etxtbsy	posix/ewouldblock
posix/exdev		

3.4 System Informations

`(posix-file-informations path)` — STklos *procedure*

Return a keyword list describing the cureent status of file "path".

```
(posix-file-informations "/")
          ⇒ (:dev 770 :ino 2 :mode 16877 :nlink 17 :uid 0 :gid 0
              :size 4096 :atime 1115017161 :mtime 1134028671
              :ctime 1134028671)
```

The following constants are defined to analyse the bits of the **mode** component returned by `posix-file-informations`:

posix/ifsock	posix/iflnk	posix/ifreg	posix/ifblk
posix/ifdir	posix/ifchr	posix/ififo	posix/isuid
posix/isgid	posix/isvtx	posix/irwxu	posix/irusr
posix/iwusr	posix/ixusr	posix/irwxg	posix/irgrp
posix/iwgrp	posix/ixgrp	posix/irwxo	posix/iroth
posix/iwoth	posix/ixoth		

`(posix-file-permissions path)` — STklos *procedure*

Return the permissions flag associated to "path" if the file is accesible. Return `#f` ostherwise.

`(posix-uid->string uid)` — STklos *procedure*

Return the login name associated to `uid`.

`(posix-uid->string logname)` — STklos *procedure*

Return the uid associated to the `logname`.

`(posix-gid->string uid)` STklos *procedure*

Return the group name associated to `gid`.

`(posix-uid->string grpname)` STklos *procedure*

Return the gid associated to the `grpname`.

`(posix-user-id)` STklos *procedure*

Return the user id of the running process.

`(posix-group-id)` STklos *procedure*

Return the group id of the running process.

`(posix-effective-user-id)` STklos *procedure*

Return the effective user id of the running process.

`(posix-effective-group-id)` STklos *procedure*

Return the effective group id of the running process.

4 Example

Here is a simple program using the STklos Posix extension. It displays some information of the path given as parameter

```
(require "stklos-posix")
(import stklos-posix)

(define (usage)
  (format (current-error-port) "Usage: ~A file\n" (program-name))
  (exit 1))

(define (bit-set? bit in)
  (= (bit-and bit in) bit))

(define (main args)
  (if (not (= (length args) 2))
    (usage)
    (let ((stat (posix-file-informations (cadr args))))
      (if (not stat)
          (format (current-error-port) "Return code ~S. Message ~S\n"
                  (posix-errno) (posix-error))
          (let ((mode (key-get stat :mode)))
            (format #t "Type of file: ~A\n"
                (cond
                  ((bit-set? posix/ifsock mode) "socket")
                  ((bit-set? posix/iflnk  mode) "symbolic link")
                  ((bit-set? posix/ifreg  mode) "regular file")
                  ((bit-set? posix/ifblk  mode) "block device")
                  ((bit-set? posix/ifchr  mode) "character device")
                  ((bit-set? posix/ifdir  mode) "directory")))
            (format #t "Owner: ~A\n"
                    (posix-uid->string (key-get stat :uid)))
            (format #t "Group: ~A\n"
                    (posix-gid->string (key-get stat :gid)))))))))
```