



e-POSIX

**The definitive and complete
Eiffel to Standard C and
POSIX 1003.1 binding**

written by Berend de Boer

Contents

1	Design notes	1
1.1	Why an entire reimplementation?	1
1.2	Goals and guidelines	1
1.3	Class structure	2
1.4	Clients of this library	3
1.5	Forking	4
2	Installation	6
2.1	Compiling the C code	6
2.1.1	Compiling on Unix	6
2.1.2	Compiling on Windows	6
2.2	Vendor specific notes	7
2.2.1	ISE Eiffel	7
2.2.2	SmallEiffel	7
2.2.3	Visual Eiffel	8
2.2.4	Halstenbach Eiffel	8
3	Basic Posix examples	9
3.1	Working with files	9
3.2	Working with file descriptors	11
3.3	Working with the file system	14
3.4	Executing a child command	17
3.5	Current time	19
3.6	Accessing environment variables	20
3.7	Allocating memory	21
4	More advanced Posix examples	22
4.1	Locking portions of files	22
4.2	Forking a child process	22
4.3	Creating a daemon	24
4.4	Talking to your modem	25
4.5	Writing CGI programs	27
4.6	Logging messages and errors	29
4.7	More examples	29
5	Standard C examples	30
5.1	Working with streams	30
5.2	Working with the file system	31
5.3	Accessing environment variables	31
5.4	Allocating memory	31
6	Accessing C headers	32
6.1	Making C Headers available to Eiffel	32
6.2	Distinction between Standard C and POSIX headers	33
6.3	C translation details	33

A	Posix function to Eiffel class mapping list	35
B	Short (flat) listing of Standard C classes	40
B.1	<i>STDC_BASE</i>	40
B.2	<i>STDC_CONSTANTS</i>	41
B.3	<i>STDC_CURRENT_PROCESS</i>	42
B.4	<i>STDC_DYNAMIC_MEMORY</i>	43
B.5	<i>STDC_ENV_VAR</i>	46
B.6	<i>STDC_FILE</i>	47
B.7	<i>STDC_FILE_SYSTEM</i>	52
B.8	<i>STDC_SYSTEM</i>	53
B.9	<i>STDC_TIME</i>	54
C	Short (flat) listing of POSIX classes	55
C.1	<i>POSIX_ASYNC_IO_REQUEST</i>	55
C.2	<i>POSIX_BASE</i>	58
C.3	<i>POSIX_CGI</i>	59
C.4	<i>POSIX_CHILD_PROCESS</i>	60
C.5	<i>POSIX_CONSTANTS</i>	61
C.6	<i>POSIX_CURRENT_PROCESS</i>	65
C.7	<i>POSIX_DAEMON</i>	66
C.8	<i>POSIX_DIRECTORY</i>	67
C.9	<i>BASE_FILE_DESCRIPTOR</i>	68
C.10	<i>POSIX_EXEC_PROCESS</i>	71
C.11	<i>POSIX_FILE_DESCRIPTOR</i>	73
C.12	<i>POSIX_FILE_SYSTEM</i>	76
C.13	<i>POSIX_FORK_ROOT</i>	79
C.14	<i>POSIX_GROUP</i>	80
C.15	<i>POSIX_LOCK</i>	81
C.16	<i>POSIX_MEMORY_MAP</i>	82
C.17	<i>POSIX_PERMISSIONS</i>	84
C.18	<i>POSIX_SIGNAL</i>	87
C.19	<i>POSIX_STATUS</i>	88
C.20	<i>POSIX_SYSTEM</i>	89
C.21	<i>POSIX_TERMIOS</i>	91
C.22	<i>POSIX_TIMED_COMMAND</i>	93
C.23	<i>POSIX_USER</i>	94
C.24	<i>XML_GENERATOR</i>	95
	To do	97
	<i>STDC_CURRENT_PROCESS</i>	97
	<i>STDC_FILE</i>	97
	<i>STDC_LOCALE_NUMERIC</i>	97
	<i>STDC_PATH</i>	97
	<i>STDC_STATUS</i>	97
	<i>STDC_STATUS</i>	97
	<i>STDC_TIME</i>	97
	<i>POSIX_CURRENT_PROCESS</i>	97

<i>POSIX_EXEC_PROCESS</i>	97
<i>POSIX_FILE_DESCRIPTOR</i>	98
<i>POSIX_MEMORY_MAP</i>	98
<i>POSIX_SEMAPHORE</i>	98
<i>POSIX_PATH</i>	98
<i>MQUEUE</i>	98
<i>DIRECTORY_BROWSER</i>	98
<i>SUS_SYSLOG</i>	98
Other	99
Known bugs	99
Bibliography	100
Index	101

Introduction

It has been a great pleasure for me when I could announce the first public alpha release of this manual. And as beta time is nearing I'm even more pleased. Writing libraries like this is boring stuff. Every Eiffel programmer should have had access to all those Standard C and POSIX routines long ago. Anyway, now you and me have. Whatever a C programmer can do, you can. And even more safe as this library protects you of inadvertently calling routines that are not portable (because they're simply not there :-)).

I will support this library, so bug reports and wishes are gladly accepted. In the future, I hope to be able to expand this library to add more stuff from the Open Unix Specification, particularly sockets and curses. Perhaps the authors of the existing Eiffel implementations for these APIs are willing to create one single unified library.

Have fun using this library and I like to hear about applications!

Licensing

This software is licensed under the Eiffel Forum Freeware License, version 1. This license can be found in the `forum.txt` file. Basically this license allows you to do anything with it, i.e. use it for commercial or Open Source software without restrictions. But don't sue me if something goes wrong. And give me some credits.

Also explicitly allowed is copying parts of this library to your own, for example copying certain Standard C or POSIX header wrappings. I prefer linking, but you don't have to retype everything if you don't want to link.

Support

e-POSIX is a fully supported program. You can send requests for help directly to me. But to help others profit from the discussion, and perhaps to get feedback when I'm short on time, it is suggested that support messages are sent to eposix@egroups.com.

Latest versions and announcements are available from <http://www.egroups.com/group/eposix>.

Commercial support

I'm available to give companies or organisations a one or two day course using POSIX and in particular this library. Prices are 1500 EUR a day, excluding VAT, travel and hotel expenses. Contact me at berend@pobox.com.

Acknowledgements

I like to thank people who, one way or another, have helped me in creating this library. They're listed in order they have been involved with this library or manual:

- **Eugene Melekhov** <eugene_melekhov@object-tools.com>: compiled it with Visual Eiffel. As Visual Eiffel is the most strict compiler, he found a great many oversights that SmallEiffel didn't catch.
- **Ida de Boer** <ida@gameren.nl>: it was she who provided you with the POSIX to Eiffel mapping table in **appendix A**.
- **Steve Harris** <scharris@worldnet.att.net>: suggested improvements, found a CAT call problem and we had an interesting discussion about forking.
- **Jörgen Tegnér** <teg@post.netlink.se> reported a problem with an example, and a bug in `POSIX_EXEC_PROCESS`.

Colophon

The text of this manual was entered with GNU Emacs 20.5.1 on RedHat Linux 6.2. It was typeset with pdf \TeX using the Con \TeX t macro package, see <http://www.pragma-ade.com>. BON diagrams were created with METAPOST.

1

Design notes

1.1 Why an entire reimplementation?

One might wonder why I reimplemented the entire Standard C and POSIX library when most vendors also have classes that deal with files, the file system, signals and such. Unfortunately, these classes are not complete nor very portable between vendors. For someone who wants to compile against all the major vendors —and there are good reasons to do this— there is currently no portable solution. That's why many portable Eiffel programs more or less contain the same code again and again. There are some attempts to write more portable libraries, for example the **Unix File/Directory Handling Cluster** by Friedrich Dominicus, but they also are not complete nor is the implementation satisfactory.

Another attempt is done by the Gobo cluster: it attempts to provide users with a set of classes that work across all Eiffel vendors by using only the native facilities offered by each implementation. This solution is also not satisfactory. I found Gobo an excellent library and I use it myself in my **xplain2sql** project, but I think its approach to portability has the following flaws:

1. Because it uses inheritance to rename classes to a common name, you might use a feature which is not available in all implementations.
2. The contract for these classes is probably not specifiable: for which platforms and which assumptions are the contracts valid? Are these contracts the same in all implementations?
3. It is still incomplete, i.e. it doesn't cover most of the POSIX routines.

That's why I started to make the entire Standard C and POSIX routines available to Eiffel programmers. All these routines are nicely wrapped in classes. I spend a lot of time designing and refactoring these, comments and improvements about its structure are very appreciated.

The advantage of making POSIX available to Eiffel programmers is that someone doesn't need to think about creating a set of portable file and directory classes that work on every known operating system. POSIX is available on many platforms and for other systems there either is an emulation or a POSIX mapping available. It's better to reuse that, instead of reinventing work that took years to complete.

1.2 Goals and guidelines

The goals and guidelines for this library were:

1. A complete Standard C implementation for those who didn't have access to POSIX routines.
2. A complete POSIX implementation.

3. Do the job in such a way that it will become the official Eiffel POSIX mapping.
4. All classes should satisfy the demands posed by the query–command separation principle.
5. The native Standard C and POSIX routines should be available to those who don't want to go through a certain class layer.
6. If a command fails, an exception code is raised. This differs from the POSIX routines where you are expected to test for error and query the `errno` variable. The only exception is `unlink`: when the file does not exist, no exception is raised.
7. POSIX assumptions should be made explicit. For Eiffel this means specifying explicit pre- and postconditions.
8. Use of constants to influence the way a method should be avoided by providing clearly named methods. So instead of passing a constants to the `POSIX_FILE.open` function to open a file read-only, you can also call `open_read`.
9. Attempt to create non-deferred class that refer to an entity that exists in the POSIX world. Creation of an object is binding to that entity, or creation of that entity.
10. Names should be clear, and Eiffel-like. They should not differ in just one character. POSIX names are also made available to ease use of this library for programmers that know POSIX well.

1.3 Class structure

e-POSIX makes available all the Standard C and POSIX headers in classes like `CAPL_STUDIO` and `PAPL_UNISTD`. You can find more details about the header translation in [chapter 6](#).

However, making the plain C API available is not a very interesting addition to an Eiffel programmer's toolkit. Therefore, this library's second attempt was to make an effective OO-wrapper mean-time making a careful distinction between what is available in the Standard C and what is available in POSIX. This distinction is reflected in e-POSIX's directory structure, see [figure 1.1](#).

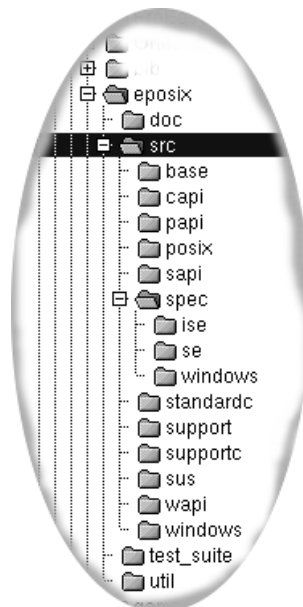


Figure 1.1 e-POSIX directory structure

The raw Standard C API is available in `src/capi`, the OO-wrapper is available in `src/standardc`. The raw POSIX API is available in `src/papi`, the OO-wrapper is available in `src/posix`.

Every Standard C and POSIX wrapper is derived from a common root, see also [figure 1.2](#):

1. Certain classes are unique to POSIX so they all inherit from [POSIX_BASE](#).
2. Certain classes are derived from or build upon facilities available in Standard C. The Standard C features are made available in classes derived from [STDC_BASE](#) and they all start with the prefix `STDC_`. POSIX classes just inherit from the Standard C classes or add new features. Such a class inherits from both [STDC_BASE](#) and [POSIX_BASE](#).
3. Certain POSIX capabilities are available in non POSIX platforms as well, for example in Microsoft Windows. A large part of the [POSIX_FILE_SYSTEM](#) and [POSIX_FILE_DESCRIPTOR](#) classes can be used on Windows too. In such cases the common functionality is abstracted so it can more easily be reused. The root of these classes is either the abstract class or [STDC_BASE](#).

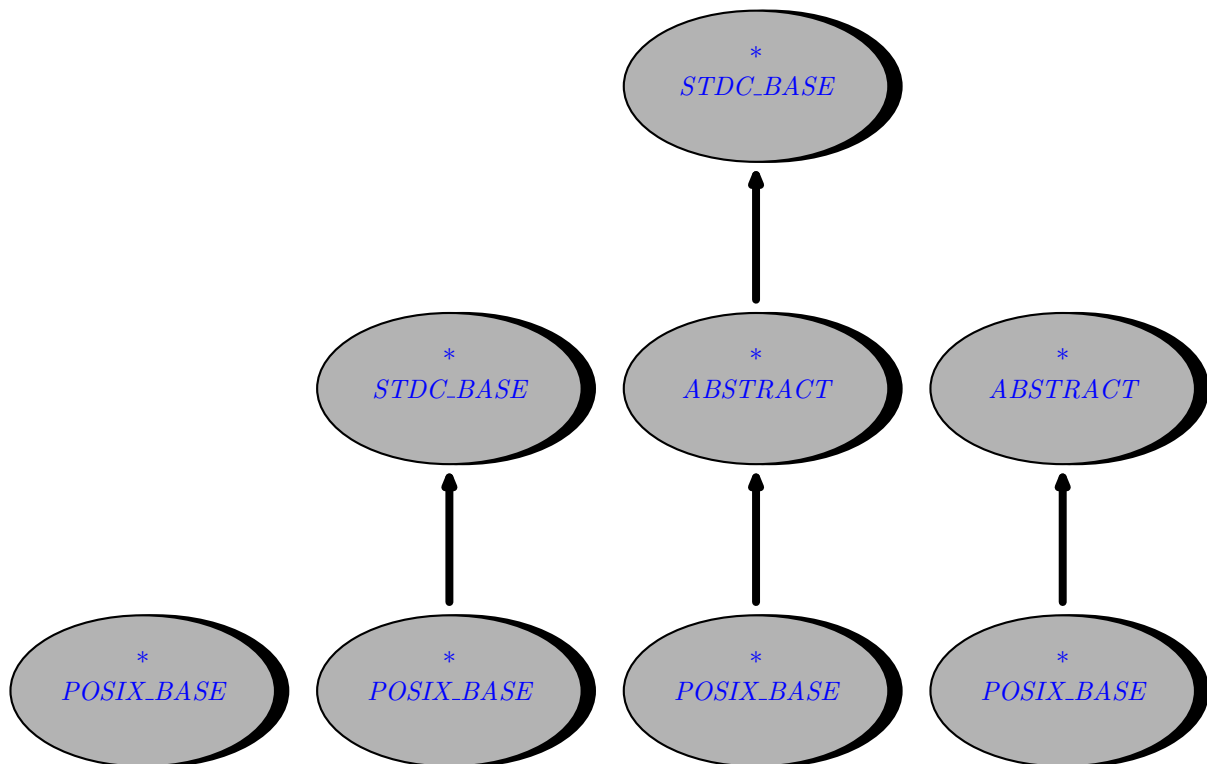


Figure 1.2 Inheritance structure

The wrapper classes should be fully command–query separated and use clear names. Often the POSIX name, if applicable, is also made available as an alias. If this is a good thing, I’m not sure. I hope it facilitates working with the wrapper classes if you already know POSIX.

1.4 Clients of this library

For client classes, two important classes are [STDC_CONSTANTS](#) and [POSIX_CONSTANTS](#), see [figure 1.3](#). The wrapper classes tend to avoid having routines whose behavior drastically depends

on passed constants. But if you need to use constants, your client class can just inherit from these classes and every Standard C and POSIX constant is available.

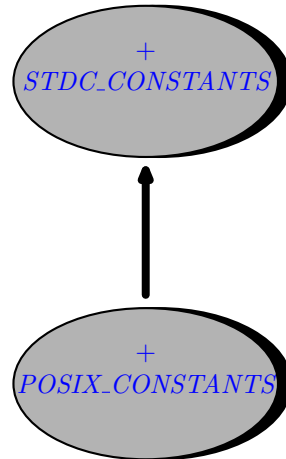


Figure 1.3 Standard C and POSIX constants

1.5 Forking

Implementing forking posed some interesting challenges. I started with the basic idea that every process has a pid:

```

class PROCESS

feature

    pid: INTEGER

end
  
```

I wanted to be able to write two kinds of forking. The first one is forking a child as in:

```

class PARENT

inherit

    POSIX_CURRENT_PROCESS

feature

    make is
    local
        child: POSIX_CHILD_PROCESS
    do
        print ("My pid: ")
        print (pid)
    end
end
  
```

```
    print ("%N")
    fork (child)
    print ("child's pid: ")
    print (child.pid)
    print ("%N")
    child.wait_for (True)
end
```

end

However, I also wanted to fork myself, because that basically is what forking is!

class PARENT

inherit

POSIX_CURRENT_PROCESS

POSIX_CHILD_PROCESS

feature

```
make is
do
    fork (Current)
    wait
end
```

```
execute is
do
    -- forked code
end
```

end

The above code gives a name clash, because *POSIX_CURRENT_PROCESS.pid* is a call to the POSIX routine `getpid`, while the child's pid is a variable, which gets a variable after forking. You can solve this name clash yourself, but it is most easy to inherit from *POSIX_FORK_ROOT*, a clash which has solved this clash already.

If you fork a child, you must wait for it. For a child process, you can use *POSIX_CHILD.wait_for*, if you fork yourself, you must use *POSIX_CURRENT_PROCESS.wait*. The variable *waited_child_pid* will be set with the pid of the child process that *wait* waited for.

2

Installation

2.1 Compiling the C code

2.1.1 Compiling on Unix

As the Eiffel to C binding is made available through C code, you have to compile this code into the object library `libposix.a` before you can use the e-POSIX classes.

This can be done with:

```
make libposix.a
```

You need GNU make as this Makefile has several features not supported by the BSD make.

2.1.2 Compiling on Windows

For Windows system, a batch file is provided to make the system. Type:

```
makelib -msc
```

to compile the C code with Microsoft's Visual C compiler. It was tested with version 6.

Type:

```
makelib -bcc
```

to compile the C code with Borland's C compiler. It was tested with the free Borland C version 5.5 compiler.

Type:

```
makelib -lcc
```

to compile the C code with elj-win32's lcc C compiler.

As I've not been able to successfully melt ISE Eiffel 4.5 with Borland C 5.5, I can't verify that this work. However, the Microsoft compiler did work with ISE Eiffel 4.5.

It seems the library generated by Borland C is not usable by VisualEiffel. The Microsoft compiler generated library perhaps work. Due to a missing `<cecil.h>` file I've not yet verified that the compiled library works with VisualEiffel.

2.2 Vendor specific notes

2.2.1 ISE Eiffel

Due to the fact that I decided to use ELKS2000, in an attempt to make something that finally could be compiled by more than one compiler, and ISE Eiffel 4.5 is not yet ELKS2000 compliant, e-POSIX doesn't work out of the box. However, I've compiled and run the Standard C test programs successfully under the following conditions:

1. I used Microsoft Windows NT 4, Service Pack 6.
2. I used the Microsoft Visual C 6.0 compiler with Service Pack 3.
3. I replaced features like *STRING.is_empty* by *empty*, and *clear* by *clear_all* and perhaps some more.

The `Ace.acefile` I've used is provided in the `test_suite` directory.

2.2.2 SmallEiffel

e-POSIX was developed using SmallEiffel -0.77 beta1 and beta4 on FreeBSD and Linux.

To successfully compile with SmallEiffel you need:

1. a correct `loadpath.se`,
2. Pass either the `libposix.alibrary` or object files to the compiler.

Compiling a class which uses e-POSIX can be done with:

```
compile MYCLASS make -L../eposix/ -leposix
```

Make sure the `-L` option points to the directory where `eposix` is located so `libposix.a` can be found.

This doesn't seem to work correctly with `elj-win32` however. It seems `lcc`'s linker doesn't understand the `-l` option. On Windows you need to pass all generated e-POSIX `.obj`'s to the compiler:

```
compile MYCLASS make p:\src\eposix\src\capi\*.obj
```

A typical SmallEiffel `loadpath.se` looks like:

```
/eposix/src/sus/
/eposix/src/posix/
/eposix/src/standardc/
/eposix/src/spec/se/
/eposix/src/sapi/
/eposix/src/papi/
/eposix/src/capi/
/eposix/src/support/
/eposix/src/base/
```

Because SmallEiffel has a tendency to provide lots of routines in its kernel classes, a bad thing in my opinion, I had to write a new *ANY*. My *ANY* renames *GENERAL.remove_file*, so I wouldn't get a conflict with *POSIX_FILE_SYSTEM.remove_file*.

There is no reason for the presence of *GENERAL.remove_file*, I expect this to be removed soon, so my *ANY* can be deleted when this has happened.

2.2.3 Visual Eiffel

e-POSIX compiles almost out of the box with Visual Eiffel 3.3 beta, which is ELKS2000 compliant. I've not been able to verify that it runs, because I missed `cecil.h`.

Because VisualEiffel does not yet support the *create* keyword, use the provided `build.ve.shscript` to replace this by the bang bang syntax.

Add a cluster with name `eposix`, pointing to the `src` directory. The provided `src/cluster.es` file will give you a correct cluster.

2.2.4 Halstenbach Eiffel

e-POSIX has not been tested with this compiler.

3

Basic Posix examples

Instead of describing every class and every feature, I decided to show short and simple examples of common ways to use the Posix library features. If you don't have Posix available, you can try to replace the POSIX prefix by STDC. Most of the time the POSIX classes are based on the STDC classes, see [chapter 5](#).

3.1 Working with files

The basic class for working with files, or streams as they are also called, is [POSIX_FILE](#). There are two kinds of files: [POSIX_TEXT_FILE](#) and [POSIX_BINARY_FILE](#). Although POSIX systems do not make a distinction between binary and text files, certain systems you can compile Posix code on do. On all variants of windows you need this distinction, even if you use the Cygwin libraries.

The first example shows how to open a text file, see also the corresponding BON diagram in [figure 3.1](#).

class *EX_FILE1*

creation

make

feature

```
make is
local
  file: POSIX_TEXT_FILE
do
  create file.open_read ("/etc/group")
  from
  until
    file.eof
  loop
    file.read_string (256)
    print (file.last_string)
  end
  file.close
end
```

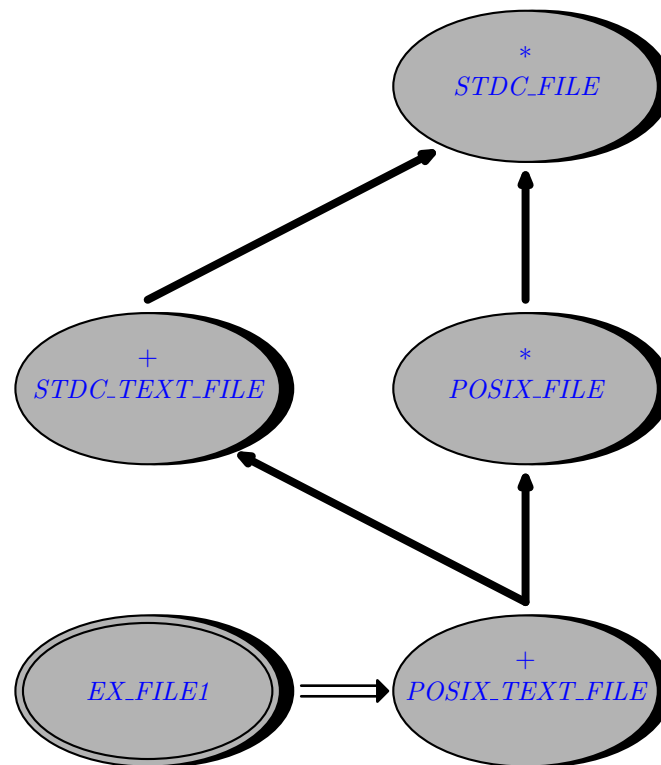


Figure 3.1 BON diagram of opening a text file.

end

It simply opens a file for reading and prints every line in it. Note that you have to specify the maximum number of characters you are prepared to read. The minimum characters read are 256, but perhaps you want to be able to read text files consisting of 1024 characters per line.

At the end the file is closed. You don't need to explicitly close a file as it will be closed when your object is garbage collected. But I think it's a good thing not to rely or depend on this, but to close your external resources as soon as you're done using them. For example many systems have easily reached limits on the number of files a process can have open.

In the second example a binary file is created and a string is written to it.

class *EX_FILE2*

creation

make

feature

make is

local

file: POSIX_BINARY_FILE

do


```
        create file.create_write ("HOME/myfile.tmp")
        file.write_string ("hello world.%N")
        file.close
    end
```

end

This example also demonstrates a nice feature that pathnames —file and directory names— have: if they contain one or more environment variables, they are expanded before the name is used. And depending on the platform you are running a backslash is turned into a slash or vice versa.

3.2 Working with file descriptors

The file descriptors classes are quite equal to the file classes. The following example opens a file using *POSIX_FILE_DESCRIPTOR* and reads the first 64 bytes.

```
class EX_FD1

    creation

        make

    feature

        make is
        local
            fd: POSIX_FILE_DESCRIPTOR
        do
            create fd.open_read ("/etc/group")
            fd.read_string (64)
            print (fd.last_string)
            fd.close
        end
    end
```

end

Unlike *POSIX_TEXT_FILE*, there is no easy way to detect end of line and end of file conditions. However, a file descriptor can easily be turned into a file as the following example demonstrates.

```
class EX_FD2

    creation

        make

    feature

        make is
        local
```

```

    fd: POSIX_FILE_DESCRIPTOR
    file: POSIX_TEXT_FILE
do
    create fd.open_read ("/etc/group")
    create file.make_from_file_descriptor (fd, "r")
from
until
    file.eof
loop
    file.read_string (256)
    print (file.last_string)
end
file.close
fd.close
end

```

end

A file descriptor can also be used to lock, unlock or test for locks on a given file as the following example demonstrates. See also the accompanying BON diagram in [figure 3.2](#).

class *EX_FD4*

creation

make

feature

```

make is
local
    lock: POSIX_LOCK
    fd: POSIX_FILE_DESCRIPTOR
do
    create fd.create_read_write ("test.tmp")
    fd.write_string ("Test")

    create lock.make
    lock.set_allow_read
    lock.set_start (2)
    lock.set_length (1)
    if fd.get_lock (lock) then
        print ("There is already a lock?%N")
    end

    lock.set_allow_none
    lock.set_start (0)
    lock.set_length (4)
    fd.set_lock (lock)

```

```

    fd.close
end

```

```

end

```

Perhaps `POSIX_FILE_DESCRIPTOR.get_lock` is not entirely command–query separated, but I couldn’t come up with a better solution. You pass a `POSIX_LOCK` to `get_lock` and it returns True if there is already a lock. The passed parameter `POSIX_LOCK` is set to the details of the lock.

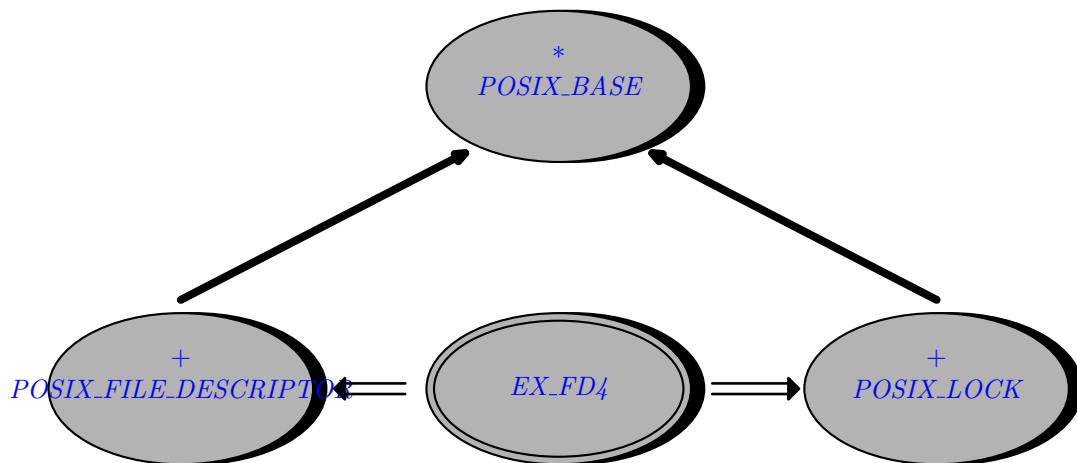


Figure 3.2 BON diagram of locking a portion of a file.

A file descriptor also gives you access to the attached terminal, if any. The following example demonstrates how to read a password without the password appearing on the screen.

```

class EX_FD3

inherit

    POSIX_CURRENT_PROCESS

creation

    make

feature

    make is
    do
        print ("Password: ")
        stdout.flush

        -- turn off echo
        fd_stdin.terminal.set_echo_input (False)
        fd_stdin.terminal.apply_flush
    end
end

```

```

-- read password
fd_stdin.read_string (256)

-- turn echo back on
fd_stdin.terminal.set_echo_input (True)
fd_stdin.terminal.apply_now

print ("%NYour password was: ")
print (fd_stdin.last_string)
end

end

```

3.3 Working with the file system

POSIX defines many commands to navigate a file system. They're made available by the *POSIX_FILE_SYSTEM*. The following example navigates to the user's home directory, create a directory and removes it.

```

class EX_DIR1

inherit

  POSIX_FILE_SYSTEM
  rename
    make as make_file_system
  end

  creation

    make

  feature

    make is
    do
      make_file_system
      change_directory ("~")
      make_directory ("qqtest.xyz.tmp")
      remove_directory ("qqtest.xyz.tmp")
    end

  end

```

To get access to the file system, inheriting from the *POSIX_FILE_SYSTEM* class is easiest. Don't forget to call the creation routine of *POSIX_FILE_SYSTEM* though.

There are also lots of functions to test for existence, readability or writability of files. Use *is_modifiable* to test if a file is readable and writable.

```
class EX_DIR2

inherit

    POSIX_FILE_SYSTEM
    rename
        make as make_file_system
    end

creation

    make

feature

    make is
        local
            perm: POSIX_PERMISSIONS
        do
            make_file_system

            print_info (is_existing ("/tmp"), "existing")
            print_info (is_executable ("/bin/ls"), "executable")
            print_info (is_readable ("/etc/passwd"), "readable")
            print_info (is_writable ("/etc/passwd"), "writable")
            print_info (is_modifiable ("/etc/passwd"), "readable and writable")

            perm := permissions("/etc/passwd")

            if perm.allow_group_read then
                print ("Group is allowed to read /etc/passwd.%N")
            else
                print ("Group is not allowed to read /etc/passwd.%N")
            end

            if perm.allow_anyone_read_write then
                print ("Anyone is allowed to read file.tmp.%N")
            else
                print ("Anyone is not allowed to read file.tmp.%N")
            end
        end
    end

    print_info (ok: BOOLEAN; what: STRING) is
        do
            print ("is_")
            print (what)
```

```

        print (" returned ")
        print (ok)
        print (".%N")
    end
end

```

end

Be aware that *POSIX_FILE_SYSTEM.is_readable* uses the real user and group IDs instead of the effective ones.

As you can see in the above example you can test for the permissions of a file using the *POSIX_PERMISSIONS* class. A new permissions class is created for every *POSIX_FILE_SYSTEM.permissions* call, so it is best to cache this object. If the permissions change on the file system, this class does not reflect reality anymore, because it caches the permissions. Use *POSIX_PERMISSIONS.refresh* to update the contents. Use *set_allow_group_write*, *set_allow_anyone_read* and such to set permissions.

e-POSIX also gives you access to the `stat` function using the *POSIX_STATUS* class.

class *EX_DIR4*

inherit

```

    POSIX_FILE_SYSTEM
    rename
        make as make_file_system
    end

```

creation

make

feature

```

    make is
    local
        stat: POSIX_STATUS
    do
        make_file_system

        stat := status ("/etc/passwd")
        print ("size: ")
        print (stat.size.out)
        print (".%N")
        print ("uid: ")
        print (stat.permissions.uid)
        print (".%N")
    end
end

```

end

The *POSIX_STAT*, and through it *POSIX_PERMISSIONS*, are also returned by *POSIX_FILE_DESCRIPTOR*. *status*.

Browsing a directory can be done by allocated a *POSIX_DIRECTORY* class through the *POSIX_FILE_SYSTEM*. *browse_directory* feature:

```

class EX_DIR3

inherit

    POSIX_FILE_SYSTEM
    rename
        make as make_file_system
    end

creation

    make

feature

    make is
    local
        dir: POSIX_DIRECTORY
    do
        make_file_system

    from
        dir := browse_directory (".")
        dir.start
    until
        dir.exhausted
    loop
        print (dir.item)
        print ("%N")
        dir.forth
    end
    dir.close
end

end

```

As can be seen, *POSIX_DIRECTORY* follows EiffelBase conventions.

3.4 Executing a child command

Any command line can be executed by using the *POSIX_SHELL_COMMAND* class. Just pass a command line and *execute* it.

```

class EX_CMD

creation

    make

feature

    make is
    local
        command: POSIX_SHELL_COMMAND
    do
        create command.make ("/bin/ls *")
        command.execute
        print ("Exit code: ")
        print (command.exit_code)
        print ("%N")
    end

end

```

Of course, unlike filenames and directory names, the passed command line is not subject to environment variable expansion by Eiffel itself. Any expansion is done by the shell to which the command is passed.

Often one wants to redirect the output of the program that is being executed. For such cases use the *POSIX_EXEC_PROCESS* class.

```

class EX_EXECI

inherit

    POSIX_CURRENT_PROCESS

creation

    make

feature

    make is
    local
        ls: POSIX_EXEC_PROCESS
    do
        -- necessary under SmallEiffel
        ignore_child_stop_signal

        -- list contents of current directory
        create ls.make_capture_output ("ls", <<"-l", ".>>)

```



```

ls.execute
print ("ls pid: ")
print (ls.pid)
print ("%N")
from
    ls.stdout.read_string (512)
until
    ls.stdout.eof
loop
    print (ls.stdout.last_string)
    ls.stdout.read_string (512)
end

-- close captured io
ls.stdout.close

-- wait for process
ls.wait_for (True)
end

```

end

Besides capturing output, you can also capture the standard input and standard error of the executed process.

3.5 Current time

e-POSIX has a very complete class to work with times. A time can be set from the current time by using *POSIX_TIME.make_from_now*. Before a time can be printed, it needs to be converted to either local time or UTC. Date and times can be printed using features as *default_format*, *local_date_string*, *local_time_string* or a custom format through *format*.

class *EX_TIME1*

creation

make

feature

make is

local

time1,

time2: POSIX_TIME

do

create *time1.make_from_now*

time1.to_local

print_time (time1)

```

time1.to_utc
print_time (time1)
create time2.make_time (0, 0, 0)
print_time (time2)
create time2.make_date_time (1970, 10, 31, 6, 55, 0)
time2.to_utc
print_time (time2)

if time2 < time1 then
    print ("time2 is less than time1 as expected.%N")
else
    print ("!! time2 is not less than time1.%N")
end
end

print_time (time: POSIX_TIME) is
do
    print ("Date: ")
    print (time.year)
    print ("-")
    print (time.month)
    print ("-")
    print (time.day)
    print (" ")
    print (time.hour)
    print (":")
    print (time.minute)
    print (":")
    print (time.second)
    print ("%N")
    print ("Weekday: ")
    print (time.weekday)
    print ("%N")
    print ("default string: ")
    print (time.default_format)
    print ("%N")
end

end

```

3.6 Accessing environment variables

With the class `POSIX_ENV_VAR`, the contents of environment variables can be queried. Unfortunately, POSIX does not define a portable function to set environment variables, but perhaps I should just add `putenv` as it is in the Single Unix Specification, so probably available on most POSIX platforms.

```
class EX_ENVI

creation

    make

feature

    make is
        local
            env: STDC_ENV_VAR
        do
            create env.make ("HOME")
            print (env.value)
            print ("%N")
        end
    end

end
```

3.7 Allocating memory

Allocating dynamic memory is very useful, but not portably available for Eiffel programmers. With *POSIX_DYNAMIC_MEMORY* memory can be allocated, read and written to.

```
class EX_MEM

creation

    make

feature

    make is
        local
            mem: POSIX_DYNAMIC_MEMORY
            byte: INTEGER
        do
            create mem.allocate (256)
            mem.poke_byte (2, 57)
            byte := mem.peek_byte (2)
            mem.resize (512)
            mem.deallocate
        end
    end

end
```

4

More advanced Posix examples

4.1 Locking portions of files

It looks like locking should work, but I've not been able to demonstrate this yet by a correct test class.

4.2 Forking a child process

Forking is very easy with this Eiffel POSIX implementation. The steps:

1. Write a child by inheriting from *POSIX_FORK_ROOT* and implementing its *execute* method.
2. The class that will do the forking, should inherit from *POSIX_CURRENT_PROCESS*.
3. Pass the child to the inherited feature *POSIX_CURRENT_PROCESS.fork* and the forking has begun.

The following class shows the process that forks the child.

class

EX_FORK1

inherit

POSIX_CURRENT_PROCESS

POSIX_FILE_SYSTEM

rename

make as make_file_system

end

creation

make

feature

make is

local

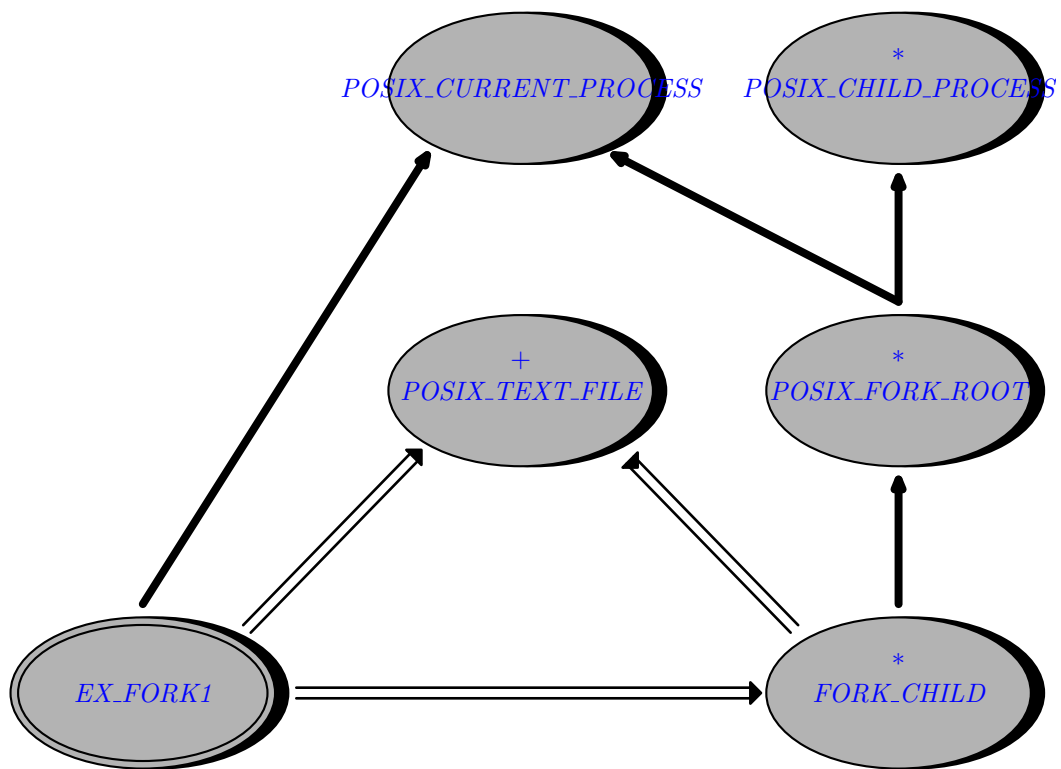


Figure 4.1 BON diagram of forking a child process.

```

reader: POSIX_TEXT_FILE
stop_sign: BOOLEAN
child: FORK_CHILD
do
  make_file_system

  -- necessary under SmallEiffel
  ignore_child_stop_signal

  unlink ("berend.tmp")
  make_fifo ("berend.tmp", S_IRUSR + S_IWUSR)
  create child
  fork (child)

  -- we will now block until file is opened for writing
  create reader.open_read ("berend.tmp")
  from
    stop_sign := False
  until
    stop_sign
  loop
    reader.read_string (128)

```

```

        print (reader.last_string)
        stop_sign := equal(reader.last_string, "stop%N")
    end
    reader.close

    -- now wait for the writer to terminate
    child.wait_for (True)

    unlink ("berend.tmp")
end

end

```

This class just displays anything that the writer, the child class, writes to the FIFO. When it recognizes stop, the reader stops after waiting for the child it has spawned. Note that this is very important! Wait for any child you have spawned else you might get spurious errors if the process exits and a child has not yet finished.

The following class shows the forked child.

```

class FORK_CHILD

inherit

    POSIX_FORK_ROOT

feature

    execute is
        local
            writer: POSIX_TEXT_FILE
        do
            create writer.open_append ("berend.tmp")
            writer.write_string ("first%N")
            writer.write_string ("stop%N")
            writer.close

            -- we give the reader some time to process these messages
            sleep (10)
        end
    end

end

```

4.3 Creating a daemon

Creating a simple daemon is easy if you inherit from *POSIX_DAEMON*. Implement the *execute* method, and you're done. At run-time, call *detach* to fork off a child. You can call *detach* as many times as you want to spawn daemons.

```
class EX_DAEMON

inherit

    POSIX_DAEMON

creation

    make

feature -- the parent

    make is
    do
        -- necessary under SmallEiffel
        ignore_child_stop_signal

        if argument_count = 0 then
            print ("Options:%N")
            print ("-d      start daemon%N")
        else
            if equal(argument(1), "-d") then
                detach
                print ("Daemon started.%N")
                print ("Its pid: ")
                print (last_child_pid)
                print ("%N")
            end
        end
    end

feature -- the daemon

    execute is
    do
        -- daemon stays alive for 20 seconds
        sleep (20)
    end

end
```

4.4 Talking to your modem

With e-POSIX you can talk to your modem. The implementation contains not all the details to write a full-featured program as minicom, but they will be added upon request.

The following example tries to talk to your modem —which is expected to be at `/dev/modem`— and queries its manufacturer.

```

class EX_MODEM

inherit

    POSIX_CURRENT_PROCESS

creation

    make

feature

    make is
    local
        modem: POSIX_FILE_DESCRIPTOR
        term: POSIX_TERMIOS
    do
        -- assume there is a /dev/modem device
        create modem.open_read_write ("/dev/modem")
        term := modem.terminal
        term.flush_input
        print ("Input speed: ")
        print (term.speed_to_baud_rate (term.input_speed))
        print ("%N")
        print ("Output speed: ")
        print (term.speed_to_baud_rate (term.output_speed))
        print ("%N")

        term.set_input_speed (B9600)
        term.set_output_speed (B9600)
        term.set_receive (True)
        term.set_echo_input (False)
        term.set_echo_new_line (False)
        term.set_input_control (True)
        term.apply_flush

        -- expect modem to echo commands
        modem.write_string ("AT%N")
        modem.read_string (64)
        print ("Command: ")
        print (modem.last_string)
        modem.read_string (64)
        print ("Response (expect ok): ")
        print (modem.last_string)
        modem.write_string ("ATIO%N")

```



```

    modem.read_string (64)
    print ("Command: ")
    print (modem.last_string)
    modem.read_string (64)
    print ("Response: ")
    print (modem.last_string)
    modem.close
end

end

```

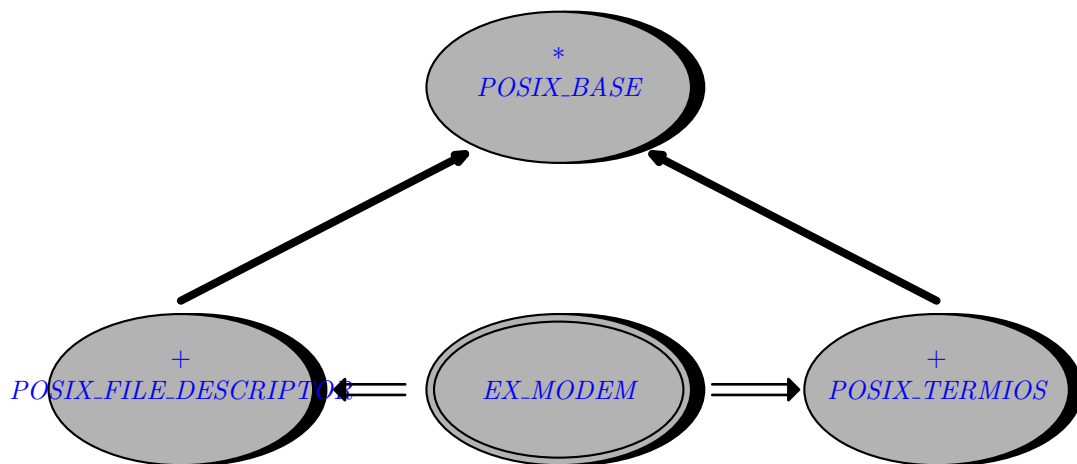


Figure 4.2 BON diagram of talking to a modem.

4.5 Writing CGI programs

Although writing a CGI program doesn't really belong to POSIX, they still written often, so I decided to include a few classes to make this easier. And of course, they build upon the standard POSIX classes.

You just inherit from `POSIX_CGI` and start calling its features.

```

class EX_CGI

inherit

    POSIX_CGI
    rename
        make as make_cgi
    end

creation

    make

```

feature

```
make is
do
    make_cgi

    content_text

    doctype
    b_html

    b_head
    title ("EPosix CGI example.")
    e_head

    b_body

    p ("Hello World.")
    add_data ("<p>you can use your <b>own</b> tags.</p>")
    b_p
    add_data ("or use any tag by using:")
    e_p

    start_tag ("table")
    set_attribute ("border", Void)
    set_attribute ("cols", "3")
    start_tag ("tr")
    start_tag ("td")
    add_data ("start_tag")
    stop_tag
    start_tag ("td")
    add_data ("stop_tag")
    stop_tag
    stop_tag
    stop_tag

    e_body
    e_html

end
```

end

It is important not to mix writing to stdout with the features you inherit from *POSIX_CGI*. *POSIX_CGI* does some caching, so after a tag is started by *POSIX_CGI.start_tag* it is not yet written to standard output. If you want to write something to standard output, use the *POSIX_CGI.add_data* feature.

4.6 Logging messages and errors

Although POSIX doesn't have logging facilities, the Single Unix Specification does. This specification requires the presence of the `syslogd` daemon for centralizes logging facilities. The following example shows you to write messages to this daemon

```
class EX_SYSLOG

inherit

    SUS_SYSLOG
    rename
        make as make_syslog
    end

creation

    make

feature

    make is
    do
        make_syslog ("test", LOG_ODELAY + LOG_PID, LOG_USER)

        debug_dump ("this is a debug message")
        info ("this is an informational message")
        warning ("this is a warning")
        error ("this is an error message")

        close
    end

end
```

Make sure there is just a single *SUS_SYSLOG* class in your system. It doesn't make sense to open a connection to the logging daemon twice.

4.7 More examples

If you are looking for more examples, you might take a look at the classes in the `test_suite` directory. These classes should demonstrate and test almost every feature available in the POSIX classes.

5

Standard C examples

If you don't have access to a POSIX compatible system, you can use the underlying Standard C classes. Standard C is quite restricted in certain respects: you cannot change directories for example. On the other hand, this library gives you access to all Standard C routines, so you can use what's there and write an extremely portable program.

All Standard C classes start with STDC_. They are:

1. *STDC_TEXT_FILE*: access text files.
2. *STDC_BINARY_FILE*: access binary files.
3. *STC_TEMPORARY_FILE*: create a temporary file, a file that is removed when it is closed or when the program terminates.
4. *STDC_CONSTANTS*: access Standard C constants like error codes and such.
5. *STDC_DYNAMIC_MEMORY*: allocate dynamic memory.
6. *STDC_ENV_VAR*: access environment variables.
7. *STDC_FILE_SYSTEM*: delete and rename files.
8. *STDC_SHELL_COMMAND*: pass an arbitrary command to the native shell.
9. *STDC_SYSTEM*: access information about the system the program is running on.
10. *STDC_CURRENT_PROCESS*: access to current process related information like its standard input, output and error streams.
11. *STDC_TIME*: access current time. Also can format a given time in various formats.

5.1 Working with streams

Working with text files is equal to the POSIX classes, only you use the STC prefix.

class *EX_FILE3*

creation

make

feature

make is

local

file: STDC_TEXT_FILE

do

```

create file.open_read ("/etc/group")
from
until
    file.eof
loop
    file.read_string (256)
    print (file.last_string)
end
    file.close
end

end

```

Its BON diagram, see figure ?? is therefore quite equal to the POSIX one, see figure ??.

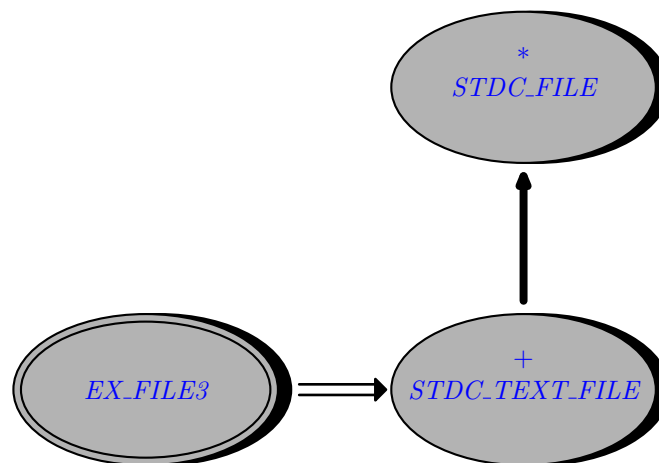


Figure 5.1 BON diagram of opening a Standard C text file.

5.2 Working with the file system

5.3 Accessing environment variables

5.4 Allocating memory

6

Accessing C headers

This chapter explains the conventions that e-POSIX uses to access the C-headers.

6.1 Making C Headers available to Eiffel

The most portable and safest header translation comes when a C function is not called verbatim, but instead a translation function is used. For example to make the Standard C function `fopen` available within Eiffel a new header file is created which lists an Eiffel compatible way to call this routine:

```
#include "eiffel.h"
#include <stdio.h>
```

```
EIF_POINTER posix_fopen(EIF_POINTER filename, EIF_POINTER mode);
```

Instead of using C types, we use Eiffel types here, which are made available by including `eiffel.h`.

The corresponding C file contains the following implementation:

```
#include "my_new_header.h"

EIF_POINTER posix_fopen(EIF_POINTER filename, EIF_POINTER mode)
{
    return ( (EIF_POINTER) fopen (filename, mode));
}
```

It simply calls the original function, returning the result. Type conversion between Eiffel and C types shouldn't pose problems this way.

To be able to call this function from Eiffel, an **external** feature needs to be written. For example:

```
class HEADER_STDIO
```

```
feature {NONE} -- C binding for stream functions
```

```
    posix_fopen (path, a_mode: POINTER): POINTER is
        -- Opens a stream
    require
        valid_mode: a_mode /= default_pointer
    external "C"
    end
```

end

Of course, the Eiffel function can have all Design By Contract features Eiffel programmers are accustomed too.

To recapitulate: every header that is to be translated, needs:

1. a new header file, and
2. a corresponding C file, and
3. an Eiffel class.

For example to translate `<stdio.h>` a header file like `eiffel_stdio.h` and a C file `eiffel_stdio.c` is needed. The Eiffel class could be in `header_stdio.e`.

6.2 Distinction between Standard C and POSIX headers

However, POSIX sometimes defines extensions to existing Standard C headers. Simply using a translation header file like `eiffel_stdio.h` will not work for pure Standard C Eiffel programs, as it can include POSIX specific extensions that might simply not be available on a given platform.

Therefore, e-POSIX divides the C headers in several groups:

1. The Standard C headers.
2. The POSIX headers.
3. The Single Unix Specification headers.
4. Microsoft Windows headers (as far as they define POSIX functions, this library does not translate Microsoft Windows specific functions).

Every group gets its own translation header with its own prefix. A translated header has a prefix, an underscore and next the original header name. The Standard C translation of `<stdio.h>` is done in `c_stdio.h` and `c_stdio.c`. The POSIX extensions to this header are available in `p_stdio.h` and `p_stdio.c`.

The corresponding Eiffel class follows similar conventions. It has the group's prefix, next the string 'API', an underscore and next the name of the header. So all `<stdio.h>` functions are made available in [*CAPL_STDIO*](#).

In [table 6.1](#) all the groups with there translation header prefix and Eiffel class prefix are listed. See also the directory structure in [figure 6.1](#).

6.3 C translation details

This translation wants to do as less as possible at the C level. It attempts to just make available the C constants and C functions and do the actual work in Eiffel.

A few details:

1. Constants, C macro definitions, are exported in the header file with the prefix 'const_' and next the macro name. The Eiffel API class exports these constants with the original, uppercased name.
2. Struct members are exported with getter and setter functions. The get function has the prefix 'posix', an underscore, the struct name, an underscore and as last the member name. The

set function has the prefix ‘posix’, an underscore, ‘set’, an underscore, the struct name, an underscore and as last the member name.

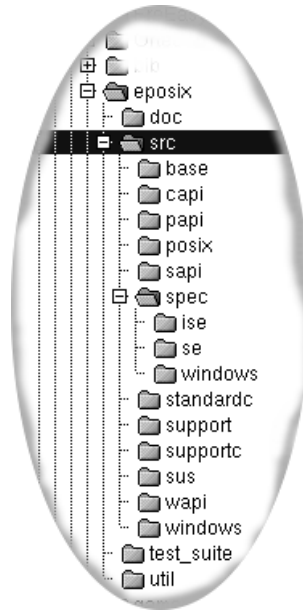


Figure 6.1 e-POSIX directory structure

Group	directory	header prefix	class prefix
Standard C	src/capi	c	CAPI
POSIX	src/[api	p	PAPI
Single Unix Specification	src/sapi	s	SAPI
Windows	src/wapi	w	WAPI

Table 6.1 hai

A

Posix function to Eiffel class mapping list

The following table defines exactly where a given Posix function is used in a Eiffel class mapping. The table is sorted in alphabetic order. Note that when a STDC_ class is listed, the feature is also available in the corresponding POSIX_ class.

Function	Header	Class	Comment
abort	<stdlib.h>	STDC_CURRENT_PROCESS.abort	
access	<unistd.h>	POSIX_FILE_SYSTEM.is_accessible	
alarm	<unistd.h>	POSIX_TIMED_COMMAND	
asctime	<time.h>	STDC_TIME.default_format	
atexit	<stdlib.h>		probably not applicable.
calloc	<stdlib.h>	STDC_DYNAMIC_MEMORY.allocate_and_clear	
cfgetispeed	<termios.h>		
cfgetospeed	<termios.h>		
cfsetispeed	<termios.h>		
cfsetospeed	<termios.h>		
chdir	<unistd.h>	POSIX_FILE_SYSTEM.change_directory	
chmod	<sys/stat.h>	POSIX_FILE_SYSTEM.change_mode	
chown	<unistd.h>	POSIX_PERMISSIONS_PATH.apply_owner_and_group	
clearerr	<stdio.h>	STDC_FILE.clear_error	
clock	<time.h>		
close	<unistd.h>	POSIX_FILE_DESCRIPTOR.close	
closedir	<dirent.h>	POSIX_DIRECTORY	
creat	<fcntl.h>	POSIX_FILE_DESCRIPTOR.create_read_write	
ctermid	<unistd.h>		
ctime	<time.h>		
cuserid	<stdio.h>		see getlogin
difftime	<time.h>	STDC_TIME	
dup	<unistd.h>	POSIX_FILE_DESCRIPTOR.make_as_duplicate	
dup2	<unistd.h>	POSIX_FILE_DESCRIPTOR.make_as_duplicate	
execl	<unistd.h>		See execvp.
execle	<unistd.h>		See execvp.
execlp	<unistd.h>		See execvp.
execv	<unistd.h>		See execvp.
execve	<unistd.h>		See execvp.
execvp	<unistd.h>	POSIX_EXEC_PROCESS.execute	
exit	<stdlib.h>	STDC_CURRENT_PROCESS.exit	
_exit	<unistd.h>		
fclose	<stdio.h>	STDC_FILE.close	
fcntl	<unistd.h>	POSIX_FILE_DESCRIPTOR	attempt_lock , get_lock , set_lock and others.

<code>fdatasync</code>	<code><unistd.h></code>	<i>POSIX_FILE_DESCRIPTOR.synchronize_data</i>	
<code>fdopen</code>	<code><stdio.h></code>	<i>POSIX_FILE.make_from_file_descriptor</i>	
<code>feof</code>	<code><stdio.h></code>	<i>STDC_FILE.eof</i>	
<code>ferror</code>	<code><stdio.h></code>	<i>STDC_FILE.error</i>	
<code>fflush</code>	<code><stdio.h></code>	<i>STDC_FILE.flush</i>	
<code>fgetc</code>	<code><stdio.h></code>	<i>STDC_FILE.get_character</i>	
<code>fgetpos</code>	<code><stdio.h></code>	<i>STDC_FILE.get_position</i>	
<code>fgets</code>	<code><stdio.h></code>	<i>STDC_FILE.get_string</i>	
<code>fileno</code>	<code><stdio.h></code>	<i>POSIX_FILE_DESCRIPTOR.make_from_file</i>	
<code>fopen</code>	<code><stdio.h></code>	<i>STDC_FILE</i>	various open creation features.
<code>fork</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.fork</i>	
<code>fpathconf</code>	<code><unistd.h></code>		
<code>fprintf</code>	<code><stdio.h></code>		not applicable.
<code>fputc</code>	<code><stdio.h></code>	<i>STDC_FILE.putc</i>	
<code>fputs</code>	<code><stdio.h></code>	<i>STDC_FILE.put_string</i>	
<code>fread</code>	<code><stdio.h></code>	<i>STDC_FILE.read</i>	Also <i>read_string</i> and <i>read_character</i> .
<code>free</code>	<code><stdlib.h></code>	<i>STDC_DYNAMIC_MEMORY.deallocate</i>	
<code>freopen</code>	<code><stdio.h></code>	<i>STDC_FILE.reopen</i>	
<code>fseek</code>	<code><stdio.h></code>	<i>STDC_FILE.seek</i>	Also <i>seek_from_current</i> and <i>seek_from_end</i> .
<code>fsetpos</code>	<code><stdio.h></code>	<i>STDC_FILE.set_position</i>	
<code>fstat</code>	<code><sys/stat.h></code>	<i>POSIX_STATUS</i>	Returned by <i>POSIX_FILE_DESCRIPTOR.status</i> .
<code>fsync</code>	<code><unistd.h></code>	<i>POSIX_FILE_DESCRIPTOR.synchronize</i>	
<code>ftell</code>	<code><stdio.h></code>	<i>STDC_FILE.tell</i>	
<code>fwrite</code>	<code><stdio.h></code>	<i>STDC_FILE.write</i>	
<code>getc</code>	<code><stdio.h></code>		Not implemented.
<code>getchar</code>	<code><stdio.h></code>		Not implemented.
<code>getcwd</code>	<code><unistd.h></code>	<i>POSIX_FILE_SYSTEM.current_directory</i>	
<code>getegid</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.effective_group_id</i>	
<code>getenv</code>	<code><stdlib.h></code>	<i>STDC_ENV_VAR.value</i>	
<code>geteuid</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.effective_user_id</i>	
<code>getgid</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.real_group_id</i>	
<code>getgrgid</code>	<code><grp.h></code>	<i>POSIX_GROUP.make_from_gid</i>	
<code>getgrnam</code>	<code><grp.h></code>	<i>POSIX_GROUP.make_from_name</i>	
<code>getgroups</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.is_in_group</i>	
<code>getlogin</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.login_name</i>	
<code>getpgrp</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.process_group_id</i>	
<code>getpid</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.pid</i>	
<code>getppid</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.parent_pid</i>	
<code>getpwnam</code>	<code><pwd.h></code>	<i>POSIX_USER.make_from_name</i>	
<code>getpwuid</code>	<code><pwd.h></code>	<i>POSIX_USER.make_from_uid</i>	
<code>gets</code>	<code><stdio.h></code>		Not implemented.
<code>getuid</code>	<code><unistd.h></code>	<i>POSIX_CURRENT_PROCESS.real_user_id</i>	
<code>gmtime</code>	<code><time.h></code>	<i>STDC_TIME.to_utc</i>	
<code>isatty</code>	<code><unistd.h></code>	<i>POSIX_FILE_DESCRIPTOR.is_attached_to_terminal</i>	
<code>kill</code>	<code><signal.h></code>	<i>POSIX_PROCESS.kill</i>	
<code>link</code>	<code><unistd.h></code>	<i>POSIX_FILE_SYSTEM.link</i>	
<code>localeconv</code>	<code><locale.h></code>	<i>STDC_LOCALE_NUMERIC</i>	
<code>localtime</code>	<code><time.h></code>	<i>STDC_TIME.to_local</i>	

lseek	<unistd.h>	<i>POSIX_FILE_DESCRIPTOR.seek</i>	Also <i>seek_from_current</i> and <i>seek_from_end</i> .
malloc	<stdlib.h>	<i>STDC_DYNAMIC_MEMORY.allocate</i>	
mblen	<stdlib.h>		
mbstowcs	<stdlib.h>		
mbtowc	<stdlib.h>		
mkdir	<sys/stat.h>	<i>POSIX_FILE_SYSTEM.make_directory</i>	
mkfifo	<sys/staat.h>	<i>POSIX_FILE_SYSTEM.make_fifo</i>	
mktime	<time.h>	<i>STDC_TIME.set_date_time</i>	Also <i>set_date</i> and <i>set_time</i> .
open	<fcntl.h>	<i>POSIX_FILE_DESCRIPTOR.open</i>	Also <i>open_read</i> , <i>open_read_write</i> and <i>open_write</i>
opendir	<dirent.h>	<i>POSIX_DIRECTORY</i>	
pathconf	<unistd.h>	<i>POSIX_DIRECTORY.max_filename_length</i>	
pause	<unistd.h>	<i>POSIX_CURRENT_PROCESS.pause</i>	
perror	<stdio.h>		
pipe	<unistd.h>	<i>POSIX_PIPE.make</i>	
printf	<stdio.h>		not applicable.
putc	<stdio.h>		
putchar	<stdio.h>		
puts	<stdio.h>		
raise	<signal.h>		
rand	<stdlib.h>		
read	<unistd.h>	<i>POSIX_FILE_DESCRIPTOR.read</i>	
readdir	<dirent.h>	<i>POSIX_DIRECTORY</i>	
realloc	<stdlib.h>	<i>STDC_DYNAMIC_MEMORY.resize</i>	
remove	<stdio.h>	<i>POSIX_FILE_SYSTEM.remove_file</i>	
rename	<unistd.h>	<i>POSIX_FILE_SYSTEM.rename_to</i>	
rewind	<stdio.h>	<i>STDC_FILE.rewind</i>	
rewinddir	<dirent.h>	<i>POSIX_DIRECTORY</i>	
rmdir	<unistd.h>	<i>POSIX_FILE_SYSTEM.remove_directory</i>	
scanf	<stdio.h>		not applicable.
setbuf	<stdio.h>	<i>STDC_FILE.set_buffer</i>	
setgid	<unistd.h>	<i>POSIX_CURRENT_PROCESS.set_group_id</i>	Also <i>restore_group_id</i> .
setlocale	<locale.h>	<i>STDC_CURRENT_PROCESS.set_locale</i>	Also <i>set_native_locale</i> and <i>set_native_time</i> .
setpgid	<unistd.h>	<i>PAPI_UNISTD.posix_setsid</i>	
setsid	<unistd.h>	<i>PAPI_UNISTD.posix_setsid</i>	
setuid	<unistd.h>	<i>POSIX_CURRENT_PROCESS.set_user_id</i>	Also <i>restore_user_id</i> .
setvbuf	<stdio.h>	<i>STDC_FILE.set_no_buffering</i>	Also <i>set_full_buffering</i> and <i>set_line_buffering</i>
sigaction	<signal.h>	<i>POSIX_SIGNAL</i>	
sigaddset	<signal.h>		
sigdelset	<signal.h>		
sigemptyset	<signal.h>		
sigfillset	<signal.h>		
sigismember	<signal.h>		
signal	<signal.h>		
sigpending	<signal.h>		
sigprocmask	<signal.h>		
sigsuspend	<signal.h>		
sleep	<unistd.h>	<i>POSIX_CURRENT_PROCESS.sleep</i>	
sprintf	<stdio.h>		Not applicable.

srand	<stdlib.h>		Applicable??
sscanf	<stdio.h>		Not applicable.
stat	<sys/stat.h>	POSIX_STATUS	
strftime	<time.h>	STDC_TIME.format	
sysconf	<unistd.h>	POSIX_SYSTEM	
system	<stdlib.h>	STDC_SHELL_COMMAND	
tcdrain	<unistd.h>		
tcflow	<unistd.h>		
tcflush	<unistd.h>		
tcgetattr	<unistd.h>	POSIX_TERMIOS.make	
tcgetpgrp	<unistd.h>		
tcsendbreak	<unistd.h>		
tcsetattr	<unistd.h>	POSIX_TERMIOS.apply_now	Also apply_drain and apply_flush
tcsetpgrp	<unistd.h>		
time	<time.h>	STDC_TIME.make_from_unix_time	
times	<times.h>		
tmpfile	<stdio.h>	STDC_TEMPORARY_FILE.make	
tmpnam	<stdio.h>	STDC_FILE_SYSTEM.temporaty_file_name	
ttyname	<unistd.h>	POSIX_FILE_DESCRIPTOR.ttyname	
tzset	<time.h>		
umask	<sys/stat.h>		
uname	<sys/utsname.h>	POSIX_SYSTEM	Various queries.
ungetc	<stdio.h>	STDC_FILE.ungetc	
unlink	<unistd.h>	POSIX_FILE_SYSTEM.unlink	
utime	<utime.h>	POSIX_FILE_SYSTEM.utime	See also its touch method.
vfprintf	<stdio.h>		Not applicable.
vprintf	<stdio.h>		Not applicable.
vsprintf	<stdio.h>		Not applicable.
wait	<sys/wait.h>	POSIX_CURRENT_PROCESS.wait	
waitpid	<sys/wait.h>	POSIX_FORK_ROOT.wait_pid	
wcstombs	<stdlib.h>		
wctomb	<stdlib.h>		
write	<unistd.h>	POSIX_FILE_DESCRIPTOR.write	

Above table contains POSIX 1003.1 functions. Most functions from POSIX 1003.1b still have to be added.

Missing:

- ctime
- fprintf
- fscanf
- pipe
- umask
- vfprintf
- sprintf

stdio: no getchar, printf, putchar, scanf, vprintf (makes sense??)

No function fgetc, but only getc. Problem? Also export fgetc?

Some signal functions also skipped for the moment.

No math/string functions.

Not <ctype.h> and <setjmp.h> and <stdarg.h>.

Not type conversion functions.

Not wide character functions.

B

Short (flat) listing of Standard C classes

B.1 STDC_BASE

```
class interface STDC_BASE
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from STDC_BASE
    -- exceptions
    raise_posix_error
    safe_call (res: INTEGER)
    -- raise an exception when value = -1
end of STDC_BASE
```

B.2 *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
feature(s) from STDC_CONSTANTS
    -- Signals
    SIGABRT: INTEGER
    SIGTERM: INTEGER
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
end of STDC_CONSTANTS
```

B.3 STDC_CURRENT_PROCESS

```

class interface STDC_CURRENT_PROCESS
feature(s) from STDC_CURRENT_PROCESS
    -- my standard input/output/error
    stdin: STDC_TEXT_FILE
    stdout: STDC_TEXT_FILE
    stderr: STDC_TEXT_FILE
feature(s) from STDC_CURRENT_PROCESS
    -- global locale
    locale: STRING
        -- return 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
    set_c_locale
        -- set locale to the Standard C locale (the default)
    set_native_decimal_point
        -- set the decimal point character
    set_native_locale
        -- set entire locale to the natives setting
    set_native_time
        -- set time display to the natives setting
end of STDC_CURRENT_PROCESS

```


B.4 STDC_DYNAMIC_MEMORY

```

class interface STDC_DYNAMIC_MEMORY
creation
  allocate (a_size: INTEGER)
    -- allocate memory of a_size bytes
    require
      valid_size: a_size > 0;
      not_allocated: not is_allocated
    ensure
      successfull_allocation: is_allocated
  allocate_and_clear (a_size: INTEGER)
    -- allocate memory of a_size bytes, make sure its zeroed out
    require
      valid_size: a_size > 0;
      not_allocated: not is_allocated
    ensure
      successfull_allocation: is_allocated
feature(s) from STDC_DYNAMIC_MEMORY
  -- public features
  allocate (a_size: INTEGER)
    -- allocate memory of a_size bytes
    require
      valid_size: a_size > 0;
      not_allocated: not is_allocated
    ensure
      successfull_allocation: is_allocated
  allocate_and_clear (a_size: INTEGER)
    -- allocate memory of a_size bytes, make sure its zeroed out
    require
      valid_size: a_size > 0;
      not_allocated: not is_allocated
    ensure
      successfull_allocation: is_allocated
  deallocate
    -- free the allocated memory now, dont wait for garbage collector.
    require
      not_deallocated: is_allocated
    ensure
      now_deallocated: not is_allocated
  resize (new_size: INTEGER)
    -- resize memory to new_size bytes. Expanded memory is not
    -- guaranteed to be zeroed out.
    require
      valid_size: new_size > 0;
      allocated: is_allocated
    ensure

```

```

    successfull_allocation: is_allocated
realloc (new_size: INTEGER)
-- resize memory to new_size bytes. Expanded memory is not
-- guaranteed to be zeroed out.
require
    valid_size: new_size > 0;
    allocated: is_allocated
ensure
    successfull_allocation: is_allocated
feature(s) from STDC_DYNAMIC_MEMORY
-- copy data from somewhere
copy_from (source: POINTER; a_size: INTEGER)
-- Copy data from source, memory may not overlap
require
    data_should_fit: a_size <= size
feature(s) from STDC_DYNAMIC_MEMORY
-- set/get bytes (8-bit data)
peek_byte (index: INTEGER): INTEGER
-- consider memory an array of 8 bit values.
require
    valid_index: index >= 0 and index < size
ensure
    possible_values: Result >= 0 and Result < 256
poke_byte (index, value: INTEGER)
require
    valid_index: index >= 0 and index < size;
    valid_value: value >= 0 and value < 256
feature(s) from STDC_DYNAMIC_MEMORY
-- set/get integers (32-bit data)
peek_integer (index: INTEGER): INTEGER
-- consider memory an array of 32 bit values.
require
    valid_index: index >= 0 and index < size // 4
feature(s) from STDC_DYNAMIC_MEMORY
-- queries
is_allocated: BOOLEAN
feature(s) from STDC_DYNAMIC_MEMORY
-- state
ptr: POINTER
-- the actual pointer
size: INTEGER
-- in number of bytes
feature(s) from STDC_DYNAMIC_MEMORY
dispose
-- Action to be executed just before garbage collection
-- reclaims an object.
invariant

```

```
valid_size: size >= 0;  
size_and_ptr_relation: (size = 0 implies not is_allocated) and size > 0 implies is_allocated;  
end of STDC_DYNAMIC_MEMORY
```

B.5 *STDC_ENV_VAR*

class *interface* *STDC_ENV_VAR*

creation

make (*a_name*: *STRING*)

require

valid_name: *a_name* /= *Void* **and then not** *a_name.is_empty* -- *a_name* doesnt have to be an existing

feature(s) from *STDC_ENV_VAR*

make (*a_name*: *STRING*)

require

valid_name: *a_name* /= *Void* **and then not** *a_name.is_empty* -- *a_name* doesnt have to be an existing

feature(s) from *STDC_ENV_VAR*

-- queries

name: *STRING*

value: *STRING*

end *of* *STDC_ENV_VAR*

B.6 STDC_FILE

deferred class interface STDC_FILE

feature(s) from STDC_FILE

-- creation

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 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

feature(s) from STDC_FILE

-- work with existing streams

attach_to_stream (*a_stream*: *POINTER*; *a_mode*: *STRING*)

-- attach to *a_stream*. Will become owner of stream so

-- it will close it when garbage collected.

require

valid_stream: *a_stream* /= Void;

valid_mode: *a_mode* /= Void **and then** *a_mode.count* > 0 -- *a_stream* is open

-- *a_mode* is compatible with *a_stream*

unattach

-- assume someone else will close this stream

feature(s) from STDC_FILE

-- close

close

ensure

closed: **not** *is_open*

feature(s) from STDC_FILE

-- reopen

reopen (*path*, *a_mode*: *STRING*)

-- closes and then opens a stream

require

open: *is_open* -- *valid_mode*: mode is a valid posix mode

-- *valid_mode*: mode is a valid posix mode

ensure

file_stays_open: *is_open*

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?
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?
set_full_buffering (buffer: POINTER; size: INTEGER)
-- Determines buffering for a stream
-- give NULL buffer so setvbuf will allocate a buffer
set_line_buffering (buffer: POINTER; size: INTEGER)
-- Determines buffering for a stream
-- give NULL buffer so setvbuf will allocate a buffer
set_no_buffering
-- Turns off buffering
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
ensure
    eof_set: last_byte = const_EOF implies eof
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
ensure
    eof_set: last_byte = const_EOF implies eof
gets (bytes: INTEGER)
-- Reads at most one less than bytes characters.
-- No additional characters are read after a newline character
-- or after end-of-file. If a newline character is read, it
-- is returned too.
-- Result is placed in last_string
get_string (bytes: INTEGER)
-- Reads at most one less than bytes characters.
-- No additional characters are read after a newline character

```

```

-- or after end-of-file. If a newline character is read, it
-- is returned too.
-- Result is placed in last_string
feature(s) from STDC_FILE
-- read, Eiffel like
last_read: INTEGER
-- last read bytes by some read_XXXX or get_string call
last_character: CHARACTER
-- last character read by getc
last_string: STRING
-- last string read by get_string
read (buf: POINTER; bytes: INTEGER)
read_character
-- read a single character and set last_character
-- if end-of-file encountered, eof is True
read_string (bytes: INTEGER)
-- Read at most n characters, a value more expected by
-- programmers not used to strings with a trailing byte.
-- result is placed in last_string
-- last_string includes the newline character!
feature(s) from STDC_FILE
-- write
last_written: INTEGER
-- last written bytes by some write_XXXX call
put (any: ANY)
-- write class as string
putc (c: INTEGER)
-- write a single character
ensure
    need_flush_set: need_flush
puts (s: STRING)
-- write a string
require
    valid_string: s /= Void
ensure
    need_flush_set: need_flush
put_string (s: STRING)
-- write a string
require
    valid_string: s /= Void
ensure
    need_flush_set: need_flush
write_string (s: STRING)
-- write a string
require
    valid_string: s /= Void
ensure

```

```

        need_flush_set: need_flush
ungetc (c: INTEGER)
    -- pushes c back to the stream
    -- note that file positioning functions discard any
    -- pushed-back characters
write (buf: POINTER; bytes: INTEGER)
    -- write bytes bytes from buf
feature(s) from STDC_FILE
    -- file position
    getpos: STDC_FILE_POSITION
    -- get the current position, use set_position to return to
    -- this saved position
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
    ensure
        not_eof: not eof;
        no_need_to_flush: not need_flush
seek (offset: INTEGER)
    -- set file position to given absolute offset
    require
        valid_offset: offset >= 0
    ensure
        not_eof: not eof;
        no_need_to_flush: not need_flush
seek_from_current (offset: INTEGER)
    -- set file position relative to current position
    ensure
        not_eof: not eof;
        no_need_to_flush: not need_flush
seek_from_end (offset: INTEGER)
    -- set file position relative to end of file
    require
        valid_offset: offset <= 0
    ensure
        not_eof: not eof;
        no_need_to_flush: not need_flush
setpos (a_position: STDC_FILE_POSITION)
    -- set the current position
    require
        valid_position: a_position /= Void
    ensure
        not_eof: not eof;

```



```

        no_need_to_flush: not need_flush
set_position (a_position: STDC_FILE_POSITION)
    -- set the current position
    require
        valid_position: a_position /= Void
    ensure
        not_eof: not eof;
        no_need_to_flush: not need_flush
tell: INTEGER
    -- The current position
feature(s) from STDC_FILE
    -- other
clearerr
    -- Clears end-of-file and error indicators for a stream
feature(s) from STDC_FILE
    -- other
clear_error
    -- Clears end-of-file and error indicators for a stream
feature(s) from STDC_FILE
    -- queries
eof: BOOLEAN
    -- True if eof encountered by getc or,
    -- if the end-of-file indicator is set
error: BOOLEAN
    -- True if and only if the error indicator is set
filename: STRING
    -- the filename of this file
is_open: BOOLEAN
mode: STRING
    -- mode in which the file is opened/created
invariant
    path_should_exist: portable_path /= Void;
    last_string_valid: last_string /= Void;
    gets_buf_valid: gets_buf /= Void;
end of deferred STDC_FILE

```

B.7 STDC_FILE_SYSTEM

```

class interface STDC_FILE_SYSTEM
creation
  make
feature(s) from STDC_FILE_SYSTEM
  -- rename files/directories, remove files/directories
  remove_file (a_path: STRING)
    -- Removes a file from a directory
    -- its not an error if this file does not exist
  require
    valid_path: a_path /= Void and then not a_path.is_empty
  require else
    a_path /= Void
  rename_to (current_path, new_path: STRING)
    -- Renames a file or directory
  require
    valid_current: current_path /= Void and then not current_path.is_empty;
    valid_new: new_path /= Void and then not new_path.is_empty
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
    -- does this by attempting to open a_path file read/write
  require
    valid_path: a_path /= Void and then not a_path.is_empty
  is_readable (a_path: STRING): BOOLEAN
    -- tests if file is readable by this program
    -- does this by attempting to open a_path file read-only
  require
    valid_path: a_path /= Void and then not a_path.is_empty
feature(s) from STDC_FILE_SYSTEM
  -- temporary names
  temporary_file_name: STRING
    -- Generates a string that is a valid non-existing file name
  ensure
    valid_name: Result /= Void and then not Result.is_empty
feature(s) from STDC_FILE_SYSTEM
  -- temporary names
  tmpnam: STRING
    -- Generates a string that is a valid non-existing file name
  ensure
    valid_name: Result /= Void and then not Result.is_empty
invariant
  path_should_exist: portable_path /= Void;
end of STDC_FILE_SYSTEM

```

B.8 *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
feature(s) from STDC_SYSTEM
  -- compile time determined queries
  clocks_per_second: INTEGER
  -- number per second of the value returned by the clock function
end of STDC_SYSTEM
```

B.9 STDC_TIME

[file stc_time.tex does not exist]

C

Short (flat) listing of POSIX classes

C.1 POSIX_ASYNC_IO_REQUEST

```
class interface POSIX_ASYNC_IO_REQUEST
creation
  make (a_fd: POSIX_FILE_DESCRIPTOR)
    require
      valid_fd: a_fd /= Void and then a_fd.is_open
feature(s) from POSIX_ASYNC_IO_REQUEST
  -- creation
  make (a_fd: POSIX_FILE_DESCRIPTOR)
    require
      valid_fd: a_fd /= Void and then a_fd.is_open
feature(s) from POSIX_ASYNC_IO_REQUEST
  -- request properties
  buffer: POINTER
    -- Location for read or written data
  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: POINTER)
    -- set buffer to read/write from
    require
      nothing_pending: not is_pending
  set_count (a_count: INTEGER)
    -- set number of bytes to read/write
    require
      nothing_pending: not is_pending
  set_offset (a_offset: INTEGER)
    require
      nothing_pending: not is_pending
feature(s) from POSIX_ASYNC_IO_REQUEST
  -- basic read/write requests
  read
```

```

-- execute async read request
require
    is_open: fd.is_open;
    nothing_pending: not is_pending
write
-- execute async write request
require
    is_open: fd.is_open;
    nothing_pending: not is_pending
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
require
    is_open: fd.is_open;
    nothing_pending: not is_pending
write_string (text: STRING)
require
    is_open: fd.is_open;
    nothing_pending: not is_pending
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
-- state
fd: POSIX_FILE_DESCRIPTOR
is_pending: BOOLEAN

```

```
-- True if io request is still pending
return_status: INTEGER
-- return status of asynchronous i/o operation, equal to what
-- the synchronous read, write of fsync would have returned
require
    nothing_pending: not is_pending
invariant
    valid_aiocb: aiocb /= Void;
end of POSIX_ASYNC_IO_REQUEST
```

C.2 *POSIX_BASE*

```
class interface POSIX_BASE  
end of POSIX_BASE
```


C.3 *POSIX_CGI*

```

class interface POSIX_CGI
feature(s) from POSIX_CGI
    -- overrule some xml stuff
    extend (stuff: STRING)
        -- add anything to the current xml string, youre on your own here!
feature(s) from POSIX_CGI
    -- cgi header
    content_text
    doctype
feature(s) from POSIX_CGI
    -- page
    b_html
    -- start html page
    e_html
    require
        valid_stop: is_started("html")
feature(s) from POSIX_CGI
    -- header
    b_head
    e_head
    require
        valid_stop: is_started("head")
    title (a_text: STRING)
feature(s) from POSIX_CGI
    -- body
    b_body
    e_body
    require
        valid_stop: is_started("body")
    b_p
    e_p
    require
        valid_stop: is_started("p")
    p (par: STRING)
invariant
    -- dont attempt to check this invariant
    -- valid_pid: pid >= 0
    same_size: attributes.count = values.count;
end of POSIX_CGI

```

C.4 *POSIX_CHILD_PROCESS*

```

deferred class interface POSIX_CHILD_PROCESS
feature(s) from POSIX_CHILD_PROCESS
  -- child's pid
  pid: INTEGER
  -- the process identifier
  require
    valid_pid: is_pid_valid
  ensure
    valid_pid: Result > 0
  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. Check the terminated property
  -- to see if this child is really terminated.
  require
    pid_refers_to_child: is_pid_valid;
    not_terminated: not is_terminated
end of deferred POSIX_CHILD_PROCESS

```

C.5 *POSIX_CONSTANTS*

```
class interface POSIX_CONSTANTS
feature(s) from POSIX_CONSTANTS
    -- error codes
    EAGAIN: INTEGER
    EBADF: INTEGER
    EINPROGRESS: INTEGER
    EINTR: INTEGER
    ENOSYS: INTEGER
feature(s) from POSIX_CONSTANTS
    -- standard file numbers
    STDERR_FILENO: INTEGER
    STDIN_FILENO: INTEGER
    STDOUT_FILENO: INTEGER
feature(s) from POSIX_CONSTANTS
    -- posix permission symbolic constants
    S_IRUSR: INTEGER
feature(s) from POSIX_CONSTANTS
    -- posix permission symbolic constants
    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 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
-- 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
B460800: INTEGER
B500000: INTEGER
B576000: INTEGER
B921600: INTEGER
B1000000: INTEGER
B1152000: INTEGER
B1500000: INTEGER
B2000000: INTEGER
B2500000: INTEGER
B3000000: INTEGER
B3500000: INTEGER
```

```
B4000000: 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
end of POSIX_CONSTANTS
```

C.6 *POSIX_CURRENT_PROCESS*

```
class interface POSIX_CURRENT_PROCESS
feature(s) from POSIX_CURRENT_PROCESS
    -- my standard input/output/error
    stdin: POSIX_TEXT_FILE
    stdout: POSIX_TEXT_FILE
    stderr: POSIX_TEXT_FILE
feature(s) from POSIX_CURRENT_PROCESS
    -- every process also has standard file descriptors
    fd_stdin: POSIX_FILE_DESCRIPTOR
    fd_stdout: POSIX_FILE_DESCRIPTOR
    fd_stderr: POSIX_FILE_DESCRIPTOR
feature(s) from POSIX_CURRENT_PROCESS
    -- POSIX locale specifics
    set_native_messages
        -- Select native language as the language in which messages
        -- are displayed
end of POSIX_CURRENT_PROCESS
```

C.7 *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
end of deferred POSIX_DAEMON
```


C.8 *POSIX_DIRECTORY*

```

class interface POSIX_DIRECTORY
creation
    make (a_directory_name: STRING)
feature(s) from POSIX_DIRECTORY
    -- creation
    make (a_directory_name: STRING)
feature(s) from POSIX_DIRECTORY
    -- access
    close
        -- close directory entry (save resources now, dont wait for
        -- garbage collection). If you call start it will automatically
        -- reopen
    require
        not_closed: is_open
    ensure
        closed: not is_open
    start
        -- start directory traversal
    forth
        -- go to next entry
    require
        opened: is_open;
        not_exhausted: not exhausted
    item: STRING
        -- the current entry
feature(s) from POSIX_DIRECTORY
    -- status report
    exhausted: BOOLEAN
        -- no more entries in this directory
    is_empty: BOOLEAN
    is_first: BOOLEAN
        -- current item is first entry
    is_open: BOOLEAN
        -- True if directory is ready for traversal
feature(s) from POSIX_DIRECTORY
    max_filename_length: INTEGER
feature(s) from POSIX_DIRECTORY
    dispose
        -- Action to be executed just before garbage collection
        -- reclaims an object.
invariant
    valid_directory_name: directory_name /= Void;
end of POSIX_DIRECTORY

```

C.9 BASE_FILE_DESCRIPTOR

Class *BASE_FILE_DESCRIPTOR* is the parent class for *POSIX_FILE_DESCRIPTOR*.

```

class interface BASE_FILE_DESCRIPTOR
feature(s) from BASE_FILE_DESCRIPTOR
  -- creation
  open (a_path: STRING; flags: INTEGER)
    -- open given file with access given by flags
    require
      closed: is_closed
  open_read (a_path: STRING)
    -- open given file with read-only access
    require
      closed: is_closed
  open_write (a_path: STRING)
    require
      closed: is_closed
  open_read_write (a_path: STRING)
    require
      closed: is_closed
  open_truncate (a_path: STRING)
    require
      closed: is_closed
  create_read_write (a_path: STRING)
    -- always create a file, existing or not
    -- give read/write permissions to user only
    require
      closed: is_closed
  create_with_mode (a_path: STRING; flags, mode: INTEGER)
    -- create a file according to flags and with mode access
    -- permissions
    require
      closed: is_closed
feature(s) from BASE_FILE_DESCRIPTOR
  -- special creation
  attach_to_fd (a_fd: INTEGER)
    -- Create file descriptor with value a_fd
    require
      closed: is_closed;
      valid_fd: a_fd >= 0 -- a_fd is open
    ensure
      opened: is_open
  make_from_file (file: STDC_FILE)
    -- Create file descriptor from given stream
    -- The stream is leading, so this file descriptor will
    -- never automatically close when garbage collected, but
    -- it will close when close is called.

```

```

-- In that case the stream is no longer valid of course,
-- but thats up to you to detect.
require
  closed: is_closed;
  valid_file: file /= Void and then file.is_open
ensure
  open: is_open
make_as_duplicate (another: BASE_FILE_DESCRIPTOR)
-- On creation, create a duplicate from another file descriptor
-- As normal call, closes its own descriptor first (if open) and
-- duplicates next.
ensure
  open: is_open
feature(s) from BASE_FILE_DESCRIPTOR
-- close
close
-- we always describe an existing object, however user probably wants
-- to have control about closing a file. And because of garbage
-- collection we cant free the file_descriptor itself.
require
  opened: is_open
ensure
  closed: is_closed
unattach
-- unbind from the current file descriptor
ensure
  closed: not is_open
feature(s) from BASE_FILE_DESCRIPTOR
-- raw read and write
last_read: INTEGER
-- last bytes read by read
read (buf: POINTER; size: INTEGER)
-- read data into buf for size bytes
require
  valid_buf: buf /= default_pointer;
  valid_size: size >= 0
write (buf: POINTER; size: INTEGER)
-- write given data
require
  valid_buf: buf /= default_pointer;
  valid_size: size >= 0
feature(s) from BASE_FILE_DESCRIPTOR
-- Eiffel like read/write
last_string: STRING
-- last read string (includes %N), see POSIX_TEXT_FILE.chop
read_string (a_size: INTEGER)
write_string (s: STRING)

```

```

feature(s) from BASE_FILE_DESCRIPTOR
  -- file position
  seek (offset: INTEGER)
    -- set file position to given absolute offset
    require
      valid_offset: offset >= 0
  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
    require
      valid_offset: offset <= 0
feature(s) from BASE_FILE_DESCRIPTOR
  -- queries
  isatty: BOOLEAN
    -- return true if handle associated with character device
feature(s) from BASE_FILE_DESCRIPTOR
  -- queries
  is_attached_to_terminal: BOOLEAN
    -- return true if handle associated with character device
  is_closed: BOOLEAN
    -- file descriptor is closed?
    ensure
      in_balance: Result implies not is_open
  is_open: BOOLEAN
    -- still describes a file descriptor?
    ensure
      in_balance: Result implies not is_closed
  status: POSIX_STATUS
  value: INTEGER
    -- return the value of the file descriptor
    require
      valid_file_descriptor: is_open
feature(s) from BASE_FILE_DESCRIPTOR
  -- accessible state
  path: STRING
invariant
  path_should_exist: portable_path /= Void;
  valid_internal_file_descriptor: fd >= - 1;
end of BASE_FILE_DESCRIPTOR

```

C.10 *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])
```

feature(s) from *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])
```

feature(s) from *POSIX_EXEC_PROCESS*

```
-- (re)set arguments
set_arguments (a_arguments: ARRAY[STRING])
```

feature(s) from *POSIX_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)
stdin: POSIX_TEXT_FILE
stdout: POSIX_TEXT_FILE
stderr: POSIX_TEXT_FILE
fd_stdin: POSIX_FILE_DESCRIPTOR
fd_stdout: POSIX_FILE_DESCRIPTOR
fd_stderr: POSIX_FILE_DESCRIPTOR
```

feature(s) from *POSIX_EXEC_PROCESS*

```
-- execute
exec
-- Executes program_name
-- dont forget to wait for this process to terminate
```

feature(s) from *POSIX_EXEC_PROCESS*

```
-- execute
execute
-- Executes program_name
-- dont forget to wait for this process to terminate
```

feature(s) from *POSIX_EXEC_PROCESS*

```
-- accessible state
program_name: POSIX_PATH
```

```
-- program to execute
arguments: ARRAY[STRING]
-- arguments to pass to program
end of POSIX_EXEC_PROCESS
```

C.11 *POSIX_FILE_DESCRIPTOR*

class *interface* *POSIX_FILE_DESCRIPTOR*

creation

open (*a_path*: *STRING*; *flags*: *INTEGER*)

-- open given file with access given by *flags*

require

closed: *is_closed*

open_read (*a_path*: *STRING*)

-- open given file with read-only access

require

closed: *is_closed*

open_write (*a_path*: *STRING*)

require

closed: *is_closed*

open_read_write (*a_path*: *STRING*)

require

closed: *is_closed*

open_truncate (*a_path*: *STRING*)

require

closed: *is_closed*

create_read_write (*a_path*: *STRING*)

-- always create a file, existing or not

-- give read/write permissions to user only

require

closed: *is_closed*

create_with_mode (*a_path*: *STRING*; *flags*, *mode*: *INTEGER*)

-- create a file according to *flags* and with *mode* access

-- permissions

require

closed: *is_closed*

make_as_duplicate (*another*: *BASE_FILE_DESCRIPTOR*)

-- On creation, create a duplicate from another file descriptor

-- As normal call, closes its own descriptor first (if open) and

-- duplicates next.

ensure

open: *is_open*

make_from_file (*file*: *STDC_FILE*)

-- Create file descriptor from given stream

-- The stream is leading, so this file descriptor will

-- never automatically close when garbage collected, but

-- it will close when *close* is called.

-- In that case the stream is no longer valid of course,

-- but thats up to you to detect.

require

closed: *is_closed*;

valid_file: *file* /= *Void* **and then** *file.is_open*

```

    ensure
        open: is_open
attach_to_fd (a_fd: INTEGER)
    -- Create file descriptor with value a_fd
    require
        closed: is_closed;
        valid_fd: a_fd >= 0 -- a_fd is open
    ensure
        opened: is_open
feature(s) from POSIX_FILE_DESCRIPTOR
    -- close
close
    -- we always describe an existing object, however user probably wants
    -- to have control about closing a file. And because of garbage
    -- collection we cant free the file_descriptor itself.
    require
        opened: is_open
    ensure
        closed: is_closed
close_on_execute
    -- close this descriptor when forking
feature(s) from POSIX_FILE_DESCRIPTOR
    -- synchronisation
synchronize
    -- synchronize the state of a file (includes synchronize_data)
    require
        synchronize_valid: supports_file_synchronization
feature(s) from POSIX_FILE_DESCRIPTOR
    -- synchronisation
fsync
    -- synchronize the state of a file (includes synchronize_data)
    require
        synchronize_valid: supports_file_synchronization
synchronize_data
    -- synchronize the data of a file
    require
        synchronize_valid: supports_synchronized_io
fdatasync
    -- synchronize the data of a file
    require
        synchronize_valid: supports_synchronized_io
feature(s) from POSIX_FILE_DESCRIPTOR
    -- locking
get_lock (a_lock: POSIX_LOCK): BOOLEAN
    -- 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

```



```
-- Test after set_lock if lock did success
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
-- queries
terminal: POSIX_TERMIOS
-- terminal settings
require
    valid_file_descriptor: is_attached_to_terminal
ensure
    valid_result: Result /= Void
ttyname: STRING
-- Terminal path name, or empty if this file descriptor does
-- not refer to a terminal
invariant
    path_should_exist: portable_path /= Void;
    valid_internal_file_descriptor: fd >= - 1;
end of POSIX_FILE_DESCRIPTOR
```

C.12 POSIX_FILE_SYSTEM

```

class interface POSIX_FILE_SYSTEM
creation
    make
feature(s) from POSIX_FILE_SYSTEM
    -- directory access
    change_directory (a_directory: STRING)
        -- Changes the current working directory
feature(s) from POSIX_FILE_SYSTEM
    -- directory access
    chdir (a_directory: STRING)
        -- Changes the current working directory
    current_directory: STRING
        -- The current directory
    getcwd: STRING
        -- The current directory
    pwd: 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 a directory
    rmdir (a_directory: STRING)
        -- Removes a directory
feature(s) from POSIX_FILE_SYSTEM
    -- read/write permissions
    chmod (a_path: STRING; a_mode: INTEGER)
        -- Changes file mode
    require
        valid_path: a_path /= Void and then not a_path.is_empty
feature(s) from POSIX_FILE_SYSTEM
    -- read/write permissions
    change_mode (a_path: STRING; a_mode: INTEGER)
        -- Changes file mode
    require
        valid_path: a_path /= Void and then not a_path.is_empty
    permissions (a_path: STRING): POSIX_PERMISSIONS
        -- return the permissions object (a new one every time!) for
        -- the given file
    require
        valid_path: a_path /= Void and then not a_path.is_empty
    set_read_only (a_path: STRING)
        -- Make given file read_only
    require

```

```

        valid_path: a_path /= Void and then not a_path.is_empty
feature(s) from POSIX_FILE_SYSTEM
-- file statistics
status (a_path: STRING): POSIX_STATUS
-- Gets information about a file
require
    valid_path: a_path /= Void and then not a_path.is_empty
touch (a_path: STRING)
-- Sets the modification and access times of a_path to the
-- current time of day.
utime (a_path: STRING; access_time, modification_time: POSIX_TIME)
-- Sets file access and modification times
feature(s) from POSIX_FILE_SYSTEM
-- accessibility of files
last_access_result: INTEGER
-- value of last access test
is_accessible (a_path: STRING; a_mode: INTEGER): BOOLEAN
-- Tests for file accessibility
access (a_path: STRING; a_mode: INTEGER): BOOLEAN
-- Tests for file accessibility
is_existing (a_path: STRING): BOOLEAN
-- 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.
require
    exists: is_existing(a_path)
is_executable (a_path: STRING): BOOLEAN
-- tests if file is executable by this program
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
require
    valid_path: a_path /= Void and then not a_path.is_empty
is_readable (a_path: STRING): BOOLEAN
-- tests if file is readable by this program
-- uses real user ID and real group ID instead of effective ones
require
    valid_path: a_path /= Void and then not a_path.is_empty
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 POSIX_FILE_SYSTEM
-- further directory access
link (existing, new: STRING)
-- Creates a hard link to a file

```

```

    require
        different_names: not existing.is_equal(new)
    unlink (a_path: STRING)
        -- Removes a directory entry (equal to remove)
        -- its not an error if path does not exist
    require
        valid_path: a_path /= Void and then not a_path.is_empty
feature(s) from POSIX_FILE_SYSTEM
    -- directory browsing
    browse_directory (a_directory: STRING): POSIX_DIRECTORY
    require
        valid_dir: a_directory /= Void and then not a_directory.is_empty
feature(s) from POSIX_FILE_SYSTEM
    -- mkfifo
    make_fifo (a_path: STRING; a_mode: INTEGER)
        -- Makes a FIFO special file
    require
        valid_path: a_path /= Void and then not a_path.is_empty
feature(s) from POSIX_FILE_SYSTEM
    -- mkfifo
    mkfifo (a_path: STRING; a_mode: INTEGER)
        -- Makes a FIFO special file
    require
        valid_path: a_path /= Void and then not a_path.is_empty
invariant
    path_should_exist: portable_path /= Void;
end of POSIX_FILE_SYSTEM

```

C.13 *POSIX_FORK_ROOT*

```

deferred class interface POSIX_FORK_ROOT
feature(s) from POSIX_FORK_ROOT
  -- process properties
  pid: INTEGER
  -- either the current process identifier or the childs
  require
    valid_pid: is_pid_valid
  ensure
    valid_pid: Result > 0
  is_valid_child_process: BOOLEAN
  -- returns True if this object seems to refer to a valid child process
  -- the real child process might have stopped though
feature(s) from POSIX_FORK_ROOT
  -- deferred routines
  execute
  -- start if child process
feature(s) from POSIX_FORK_ROOT
  -- termination info
  is_terminated_normally: BOOLEAN
  -- has this process been terminated normally
  require
    valid_status_info: is_terminated
feature(s) from POSIX_FORK_ROOT
  -- termination info
  is_exited: BOOLEAN
  -- has this process been terminated normally
  require
    valid_status_info: is_terminated
  exit_code: INTEGER
  -- low-order 8 bits of call to _exit or exit for this process
  require
    terminated_normally: is_terminated_normally
  require else
    valid_status_info: is_terminated
  is_signaled: BOOLEAN
  -- child process was terminated due to receipt of a signal
  -- that was not caught
  require
    valid_status_info: is_terminated
  signal_code: INTEGER
  -- signal of process terminated abnormally or was stopped
  require
    valid_status_info: is_terminated;
    terminated_by_signal: is_signaled
end of deferred POSIX_FORK_ROOT

```

C.14 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
    valid_group: group /= default_pointer;
end of POSIX_GROUP
```

C.15 *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
    valid_buf: buf /= Void;
    lock_type_known: allow_all or else allow_none or else allow_read;
end of POSIX_LOCK

```

C.16 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*: *INT*)
 -- raw interface to mmap

make_private (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset*, *a_size*: *INTEGER*)

-- make a mapping where changes are private
 -- 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 a mapping where changes are shared, i.e. the
 -- 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*

-- creation

make (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset*, *a_size*: *INTEGER*; *a_base*: *POINTER*; *a_prot*, *a_flags*: *INT*)
 -- raw interface to mmap

make_private (*a_fd*: *POSIX_FILE_DESCRIPTOR*; *a_offset*, *a_size*: *INTEGER*)

-- make a mapping where changes are private
 -- 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 a mapping where changes are shared, i.e. the
 -- 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*

-- reading from the map

peek_byte (*index*: *INTEGER*): *INTEGER*

-- consider memory an array of 8 bit values.

require

valid_index: *index* >= 0 **and** *index* < *size*

ensure

possible_values: *Result* >= 0 **and** *Result* < 256

feature(s) from *POSIX_MEMORY_MAP*

-- state

base: *POINTER*

-- base address

offset: *INTEGER*

-- offset from file

size: *INTEGER*


```
-- number of bytes mapping
fd: POSIX_FILE_DESCRIPTOR
end of POSIX_MEMORY_MAP
```

C.17 *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
    ensure
        executability: not allow or allow_anyone_execute
set_allow_anyone_read (allow: BOOLEAN)
    -- give anyone read permission
    ensure
        readability: not allow or allow_anyone_read
set_allow_anyone_read_write (allow: BOOLEAN)
    -- give anyone read and write permissions
    ensure
        writability: not allow or allow_anyone_read_write
set_allow_anyone_write (allow: BOOLEAN)
    -- give anyone write permission
    ensure
        writability: not allow or allow_anyone_write
set_allow_group_execute (allow: BOOLEAN)
    -- give group execute permission
    ensure
        executability: not allow or allow_group_execute
set_allow_group_read (allow: BOOLEAN)
    -- give group read permission
    ensure
        readability: not allow or allow_group_read
set_allow_group_read_write (allow: BOOLEAN)
    -- give group read and write permission
    ensure
        writability: not allow or allow_group_read_write
set_allow_group_write (allow: BOOLEAN)
    -- give group write permission
    ensure
        writability: not allow or allow_group_write
set_allow_owner_execute (allow: BOOLEAN)
    -- give owner execute permission
    ensure
        executability: not allow or allow_owner_execute
set_allow_read (allow: BOOLEAN)
    -- give read permission
    ensure
        readability: not allow or allow_owner_read
set_allow_owner_read (allow: BOOLEAN)
    -- give read permission
    ensure
        readability: not allow or allow_owner_read
set_allow_read_write (allow: BOOLEAN)
    -- give read/write permission
```

```

    ensure
        writability: not allow or allow_owner_read_write
    set_allow_owner_write (allow: BOOLEAN)
        -- give read/write permission
    ensure
        writability: not allow or allow_owner_read_write
    set_allow_write (allow: BOOLEAN)
        -- give write permission
    ensure
        writability: not allow or allow_owner_write
feature(s) from POSIX_PERMISSIONS
    -- direct access to Unix fields
    uid: INTEGER
        -- id of object owner, always 0 on NT
feature(s) from POSIX_PERMISSIONS
    -- direct access to Unix fields
    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
end of deferred POSIX_PERMISSIONS

```

C.18 *POSIX_SIGNAL*

```

class interface POSIX_SIGNAL
creation
    make (a_value: INTEGER)
feature(s) from POSIX_SIGNAL
    -- creation
    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
    set_ignore_action
        -- ignore signal
    require
        not_sig_child: value /= SIGCHLD
    set_mask (a_mask: POSIX_SIGNAL_SET)
feature(s) from POSIX_SIGNAL
    -- signal state
    child_stop: BOOLEAN
        -- generate SIGCHLD when children stop
    is_defaulted: BOOLEAN
        -- signal is handled by its specific default action
    handler: POINTER
        -- pointer to function which catches this signal
    is_ignored: BOOLEAN
        -- signal is ignored
    mask: POSIX_SIGNAL_SET
    value: INTEGER
        -- the signal
    refresh
        -- get latest state for this signal
invariant
    has_memory: sigaction /= Void;
end of POSIX_SIGNAL

```

C.19 POSIX_STATUS

```

deferred class interface POSIX_STATUS
feature(s) from POSIX_STATUS
  refresh
    -- refresh the cached information
feature(s) from POSIX_STATUS
  -- stat members
  ino: INTEGER
feature(s) from POSIX_STATUS
  -- stat members
  inode: INTEGER
  mtime: INTEGER
    -- Unix time
  modification_time: INTEGER
    -- Unix time
  change_date: INTEGER
    -- Unix time
  permissions: POSIX_PERMISSIONS
    -- file permissions
  ensure
    valid_result: Result /= Void
  size: INTEGER
    -- size of file in bytes
feature(s) from POSIX_STATUS
  -- direct access to the unix fields, not recommended
  unix_gid: INTEGER
  unix_mode: INTEGER
  unix_uid: INTEGER
invariant
  valid_stat: stat /= Void;
end of deferred POSIX_STATUS

```

C.20 *POSIX_SYSTEM*

```

class interface POSIX_SYSTEM
feature(s) from POSIX_SYSTEM
    -- sysconf queries, run-time determined
    arg_max: INTEGER
        -- The length of arguments for the exec() function
    child_max: INTEGER
        -- The number of simultaneous processes per real user ID
    clock_ticks: INTEGER
        -- The number of clock ticks per second
    ngroups_max: INTEGER
        -- The number of simultaneous supplementary group IDs
    stream_max: INTEGER
        -- The maximum number of streams that one process can have
        -- open at one time.
    tzname_max: INTEGER
        -- The maximum number of bytes in a timezone name.
    open_max: INTEGER
        -- The maximum number of files that one process can have
        -- open at one time.
    page_size: INTEGER
        -- granularity in bytes of memory mapping and process memory locking
    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
    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
feature(s) from POSIX_SYSTEM
-- uname queries
system_name: STRING
node_name: STRING
release: STRING
version: STRING
machine: STRING
end of POSIX_SYSTEM
```


C.21 *POSIX_TERMIOS*

```

class interface POSIX_TERMIOS
creation
    make (a_fd: POSIX_FILE_DESCRIPTOR)
        require
            valid_file_descriptor: a_fd.is_attached_to_terminal
feature(s) from POSIX_TERMIOS
    -- creation
    make (a_fd: POSIX_FILE_DESCRIPTOR)
        require
            valid_file_descriptor: a_fd.is_attached_to_terminal
feature(s) from POSIX_TERMIOS
    -- 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
        -- returns terminal input baud rate as symbolic value
    output_speed: INTEGER
        -- returns terminal output baud rate as symbolic value

```

```

    set_input_speed (new_rate: INTEGER)
        -- sets terminal input baud rate, new_rate is one of the
        -- BXXXX constants
    set_output_speed (new_rate: INTEGER)
        -- sets 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
    -- state
    fd: POSIX_FILE_DESCRIPTOR
        -- the file descriptor for these terminal settings
invariant
    valid_attr: attr /= Void and then attr.size = posix_termios_size;
    valid_fd: fd /= Void;
end of POSIX_TERMIOS

```

C.22 *POSIX_TIMED_COMMAND*

```
deferred class interface POSIX_TIMED_COMMAND
feature(s) from POSIX_TIMED_COMMAND
  -- creation
  make (a_seconds: INTEGER)
    require
      valid_seconds: a_seconds >= 1 and a_seconds <= 65535
feature(s) from POSIX_TIMED_COMMAND
  -- execution
  execute: BOOLEAN
    -- Return true if do_execute completed within the time it
    -- should execute.
feature(s) from POSIX_TIMED_COMMAND
  -- state
  seconds: INTEGER
    -- the number of seconds available to execute the command
  set_seconds (a_seconds: INTEGER)
invariant
  valid_seconds: seconds >= 1;
end of deferred POSIX_TIMED_COMMAND
```

C.23 *POSIX_USER*

```

class interface POSIX_USER
creation
    make_from_name (a_name: STRING)
        require
            valid_name: a_name /= Void and then not a_name.is_empty
    make_from_uid (a_uid: INTEGER)
        require
            valid_uid: a_uid >= 0
feature(s) from POSIX_USER
    -- creation
    make_from_name (a_name: STRING)
        require
            valid_name: a_name /= Void and then not a_name.is_empty
    make_from_uid (a_uid: INTEGER)
        require
            valid_uid: a_uid >= 0
feature(s) from POSIX_USER
    -- refresh cache
    refresh
        -- refresh cache with latest info from user database
feature(s) from POSIX_USER
    -- queries
    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
    valid_passwd: passwd /= default_pointer;
end of POSIX_USER

```

C.24 XML_GENERATOR

```

class interface XML_GENERATOR
creation
    make
feature(s) from XML_GENERATOR
    -- creation
    make
feature(s) from XML_GENERATOR
    -- constants from the XML specification, should be Unicode...
    ValidFirstChars: STRING
        -- which characters are valid as the first character
    ValidOtherChars: STRING
        -- which characters are valid as second etc characters
feature(s) from XML_GENERATOR
    -- queries
    is_header_written: BOOLEAN
    is_started (tag: STRING): BOOLEAN
    is_tag_started: BOOLEAN
    is_valid_attribute_name (attribute: STRING): BOOLEAN
        -- Return True if this is a valid attribute name
    xml: STRING
        -- the result
feature(s) from XML_GENERATOR
    -- influence state
    clear
        -- start fresh
    ensure
        no_tags: is_empty
feature(s) from XML_GENERATOR
    -- commands that expand xml
    add_header
        require
            valid_point_for_header: not is_header_written
    add_data (data: STRING)
        -- write data in the current tag
        require
            valid_point_for_data: is_tag_started
    add_tag (tag, data: STRING)
        -- shortcut for add_tag, add_data and stop_tag
        require
            have_header: is_header_written
    extend (stuff: STRING)
        -- add anything to the current xml string, youre on your own here!
    new_line
    set_attribute (attribute, value: STRING)
        -- set an attribute of the current tag

```

```
    require
      valid_attribute: is_valid_attribute_name(attribute)
    start_tag (tag: STRING)
      -- start a new tag
    stop_tag
      -- stop last started tag
    require
      tag_is_started: is_tag_started
  invariant
    same_size: attributes.count = values.count;
end of XML_GENERATOR
```

To do

STDC_CURRENT_PROCESS

1. Add `clock`.

STDC_FILE

1. add `read_integer`, `read_double`, `read_boolean`, etc.

STDC_LOCALE_NUMERIC

1. Complete the list of properties

STDC_PATH

1. assumes there is an access routine, not in Standard C.

STDC_STATUS

Create this class?

STDC_STATUS

1. return `STDC_TIME` instead of unix time

STDC_TIME

1. Add elapsed seconds

POSIX_CURRENT_PROCESS

1. Add `pause`.

POSIX_EXEC_PROCESS

1. turn off Eiffel exception handling after the final `execvp`, else you get back signals not captured by child process as your signals, or so it seems (or perhaps you're killing the Eiffel process, but not the subprocess it generated??)

Killing subprocesses works sometimes, but not always.

Remove exception handling just before execvp?

2. how about capture to /dev/null?
3. can we capture i/o for every forked process? If so, move this code to POSIX_FORK_ROOT.

POSIX_FILE_DESCRIPTOR

1. possible to open exclusively and so?
2. add nonblocking io
3. Asynchronous I/O: create separate class, as locking, a request to do something and pass this to the file descriptor.

POSIX_MEMORY_MAP

1. More read functions.
2. No write functions yet.
3. Cannot change protection.
4. No locking.

POSIX_SEMAPHORE

1. not valid for named semaphore I think.
2. have to add various close/unlink functions.

POSIX_PATH

1. Implement is_portable

MQUEUE

1. Not in the free unices at this moment. Maybe have to get a copy of Solaris x86??

DIRECTORY_BROWSER

1. recursive browsing
2. add filter properties

SUS_SYSLOG

1. Really is a singleton, make creation and close routines once routines? Factory?

Other

1. remove ugly `const_` prefix from constants. Uppercase should be good enough.
Almost done, only `const_EOF` remains, not easy to replace perhaps.
2. Compare `POSIX_SIGNAL` with ISE `UNIX_SIGNAL`: perhaps name routines `is_ignored`, `is_defaulted`?
They have an `is_caught` function, useful? Means this signal generates an exception. Also they can ask if it is caught.

Known bugs

- not for every `raise_posix_error` the error code is set probably.
- does `STRING_HELPER` leak memory in `to_external`? How is memory used for these conversions being freed? Is memory used there?
- If a child process is signalled (terminated), the function *[POSIX_FORK_ROOT.is_terminated_normally](#)* sometimes returns `True`.

Bibliography

Index

`_exit` 35

a

`abort` 35

abort

`STDC_CURRENT_PROCESS` 35

`access` 35

`Ace.ace` 7

add_data

`POSIX_CGI` 28

`alarm` 35

allocate

`STDC_DYNAMIC_MEMORY` 37

allocate_and_clear

`STDC_DYNAMIC_MEMORY` 35

`ANY` 7, 8

apply_drain

`POSIX_FORK_ROOT` 38

apply_flush

`POSIX_FORK_ROOT` 38

apply_now

`POSIX_TERMIOS` 38

apply_owner_and_group

`POSIX_PERMISSIONS_PATH` 35

`asctime` 35

`atexit` 35

attempt_lock

`POSIX_FORK_ROOT` 35

b

`backslash` 11

`BASE_FILE_DESCRIPTOR` ii, 68, 70

`Borland C compiler` 6

browse

`directory` 17

browse_directory

`POSIX_FILE_SYSTEM` 17

`build.ve.sh` 8

c

`c_stdio.c` 33

`c_stdio.h` 33

`calloc` 35

`CAPI_STDIO` 2, 33

`<cecil.h>` 6

`cecil.h` 8

`cfgetispeed` 35

`cfgetospeed` 35

`cfsetispeed` 35

`cfsetospeed` 35

`cgi` 27

change

`directory` 14

change_directory

`POSIX_FILE_SYSTEM` 35

change_mode

`POSIX_FILE_SYSTEM` 35

`chdir` 35

`chmod` 35

`chown` 35

clear

`POSIX_FORK_ROOT` 7

clear_all

`POSIX_FORK_ROOT` 7

clear_error

`STDC_FILE` 35

`clearerr` 35

`clock` 35, 97

`close` 35

close

`POSIX_FILE_DESCRIPTOR` 35

`STDC_FILE` 35

`closedir` 35

`creat` 35

create

`directory` 14

create_read_write

`POSIX_FILE_DESCRIPTOR` 35

`ctermid` 35

`ctime` 35

`<ctype.h>` 39

current_directory

`POSIX_FILE_SYSTEM` 36

`cuserid` 35

d*deallocate*

STDC_DYNAMIC_MEMORY 36

default_format

POSIX_FORK_ROOT 19

STDC_TIME 35

detach

POSIX_FORK_ROOT 24

difftime 35*directory*

test_suite 29

DIRECTORY_BROWSER iii, 98

<dirent.h> 35, 37

dup 35*dup2* 35**e***effective_group_id*

POSIX_CURRENT_PROCESS 36

effective_user_id

POSIX_CURRENT_PROCESS 36

eiffel.h 32*elj-win32* 7*empty*

POSIX_FORK_ROOT 7

eof

STDC_FILE 36

errno 2*error*

STDC_FILE 36

execl 35*execle* 35*execvp* 35*execute*

POSIX_FORK_ROOT 17, 22, 24

POSIX_EXEC_PROCESS 35

execv 35*execve* 35*execvp* 35*exit* 35*exit*

STDC_CURRENT_PROCESS 35

f*fclose* 35*fcntl* 35

<fcntl.h> 35, 37

fdatasync 36*fdopen* 36*feof* 36*ferror* 36*fflush* 36*fgetc* 36*fgetpos* 36*fgets* 36*fileno* 36*flush*

STDC_FILE 36

fopen 32, 36*fork* 36*fork*

POSIX_CURRENT_PROCESS 22, 36

format

POSIX_FORK_ROOT 19

STDC_TIME 38

forum.txt iv*fpathconf* 36*fprintf* 36*fputc* 36*fputs* 36*fread* 36*free* 36*freopen* 36*fseek* 36*fsetpos* 36*fstat* 36*fsync* 36*ftell* 36*fwrite* 36**g***get_character*

STDC_FILE 36

get_lock

POSIX_FORK_ROOT 13, 35

POSIX_FILE_DESCRIPTOR 13

get_position

STDC_FILE 36

get_string

STDC_FILE 36

getc 36*getchar* 36*getcwd* 36*getegid* 36

- getenv 36
- geteuid 36
- getgid 36
- getgrgid 36
- getgrnam 36
- getgroups 36
- getlogin 35, 36
- getpgrp 36
- getpid 5, 36
- getppid 36
- getpwnam 36
- getpwuid 36
- gets 36
- getuid 36
- gmtime 36
- <grp.h> 36
- i**
- is_accessible*
 - POSIX_FILE_SYSTEM 35
- is_attached_to_terminal*
 - POSIX_FILE_DESCRIPTOR 36
- is_empty*
 - STRING 7
- is_in_group*
 - POSIX_CURRENT_PROCESS 36
- is_modifiable*
 - POSIX_FORK_ROOT 14
- is_readable*
 - POSIX_FILE_SYSTEM 16
- is_terminated_normally*
 - POSIX_FORK_ROOT 99
- isatty* 36
- k**
- kill 36
- kill
 - POSIX_PROCESS 36
- l**
- lcc 6
 - linker 7
- libposix.a 6, 7
- license iv
- link 36
- link
 - POSIX_FILE_SYSTEM 36
- loadpath.se 7
- local_date_string*
 - POSIX_FORK_ROOT 19
- local_time_string*
 - POSIX_FORK_ROOT 19
- <locale.h> 36, 37
- localeconv 36
- localtime 36
- login_name*
 - POSIX_CURRENT_PROCESS 36
- lseek 37
- m**
- make*
 - POSIX_PIPE 37
 - POSIX_TERMIOS 38
 - STDC_TEMPORARY_FILE 38
- make_as_duplicate*
 - POSIX_FILE_DESCRIPTOR 35
- make_directory*
 - POSIX_FILE_SYSTEM 37
- make_fifo*
 - POSIX_FILE_SYSTEM 37
- make_from_file*
 - POSIX_FILE_DESCRIPTOR 36
- make_from_file_descriptor*
 - POSIX_FILE 36
- make_from_gid*
 - POSIX_GROUP 36
- make_from_name*
 - POSIX_GROUP 36
 - POSIX_USER 36
- make_from_now*
 - POSIX_TIME 19
- make_from_uid*
 - POSIX_USER 36
- make_from_unix_time*
 - STDC_TIME 38
- malloc 37
- max_filename_length*
 - POSIX_DIRECTORY 37
- mblen 37
- mbstowcs 37
- mbtowc 37
- Microsoft C compiler 6
- Microsoft Windows 7
- minicom 25

mkdir 37
mkfifo 37
mktime 37
modem 25
MQQUEUE iii, 98

o
open 37
open
 POSIX_FILE 2
 POSIX_FILE_DESCRIPTOR 37
open_read
 POSIX_FORK_ROOT 2, 37
open_read_write
 POSIX_FORK_ROOT 37
open_write
 POSIX_FORK_ROOT 37
opendir 37
Open Source iv

p
p_stdio.c 33
p_stdio.h 33
PAPI_UNISTD 2
parent_pid
 POSIX_CURRENT_PROCESS 36
pathconf 37
pause 37, 97
pause
 POSIX_CURRENT_PROCESS 37
permissions
 POSIX_FILE_SYSTEM 16
perror 37
pid
 POSIX_CURRENT_PROCESS 5, 36
pipe 37
POSIX_ASYNC_IO_REQUEST ii, 55, 56
POSIX_BASE ii, 3, 58
POSIX_BINARY_FILE 9
POSIX_CGI ii, 27, 28, 59
POSIX_CHILD_PROCESS ii, 60
POSIX_CONSTANTS ii, 3, 61, 62, 64
POSIX_CURRENT_PROCESS ii, 22, 65, 97
POSIX_DAEMON ii, 24, 66
POSIX_DIRECTORY ii, 17, 35, 37, 67
POSIX_DYNAMIC_MEMORY 21
POSIX_ENV_VAR 20
POSIX_EXEC_PROCESS ii, iii, 18, 71, 72, 97
POSIX_FILE 9
POSIX_FILE_DESCRIPTOR ii, iii, 3, 11, 35, 68, 73, 74, 98, 98
POSIX_FILE_SYSTEM ii, 3, 14, 76, 78
POSIX_FORK_ROOT ii, 5, 22, 79
POSIX_GROUP ii, 80
POSIX_LOCK ii, 13, 81
POSIX_MEMORY_MAP ii, iii, 82, 98
POSIX_PATH iii, 98
POSIX_PERMISSIONS ii, 16, 17, 84, 86
POSIX_SEMAPHORE iii, 98
posix_setsid
 PAPI_UNISTD 37
POSIX_SHELL_COMMAND 17
POSIX_SIGNAL ii, 37, 87
POSIX_STAT 17
POSIX_STATUS ii, 16, 36, 38, 88
POSIX_SYSTEM ii, 38, 89, 90
POSIX_TERMIOS ii, 91, 92
POSIX_TEXT_FILE 9, 11
POSIX_TIMED_COMMAND ii, 35, 93
POSIX_USER ii, 94
printf 37
process_group_id
 POSIX_CURRENT_PROCESS 36
put_string
 STDC_FILE 36
putc 37
putc
 STDC_FILE 36
putchar 37
putenv 20
puts 37
<pwd.h> 36

r
raise 37
rand 37
read 37
read
 POSIX_FILE_DESCRIPTOR 37
 STDC_FILE 36
read_character
 POSIX_FORK_ROOT 36

- read_string*
 - POSIX_FORK_ROOT 36
- readdir* 37
- real_group_id*
 - POSIX_CURRENT_PROCESS 36
- real_user_id*
 - POSIX_CURRENT_PROCESS 36
- realloc* 37
- refresh*
 - POSIX_PERMISSIONS 16
- remove* 37
- remove*
 - directory 14
- remove_directory*
 - POSIX_FILE_SYSTEM 37
- remove_file*
 - GENERAL 7, 8
 - POSIX_FILE_SYSTEM 7, 37
- rename* 37
- rename_to*
 - POSIX_FILE_SYSTEM 37
- reopen*
 - STDC_FILE 36
- resize*
 - STDC_DYNAMIC_MEMORY 37
- restore_group_id*
 - POSIX_FORK_ROOT 37
- restore_user_id*
 - POSIX_FORK_ROOT 37
- rewind* 37
- rewind*
 - STDC_FILE 37
- rewinddir* 37
- rmdir* 37
- s**
- scanf* 37
- seek*
 - POSIX_FILE_DESCRIPTOR 37
 - STDC_FILE 36
- seek_from_current*
 - POSIX_FORK_ROOT 36, 37
- seek_from_end*
 - POSIX_FORK_ROOT 36, 37
- set_allow_anyone_read*
 - POSIX_FORK_ROOT 16
- set_allow_group_write*
 - POSIX_FORK_ROOT 16
- set_buffer*
 - STDC_FILE 37
- set_date*
 - POSIX_FORK_ROOT 37
- set_date_time*
 - STDC_TIME 37
- set_full_buffering*
 - POSIX_FORK_ROOT 37
- set_group_id*
 - POSIX_CURRENT_PROCESS 37
- set_line_buffering*
 - POSIX_FORK_ROOT 37
- set_locale*
 - STDC_CURRENT_PROCESS 37
- set_lock*
 - POSIX_FORK_ROOT 35
- set_native_locale*
 - POSIX_FORK_ROOT 37
- set_native_time*
 - POSIX_FORK_ROOT 37
- set_no_buffering*
 - STDC_FILE 37
- set_position*
 - STDC_FILE 36
- set_time*
 - POSIX_FORK_ROOT 37
- set_user_id*
 - POSIX_CURRENT_PROCESS 37
- setbuf* 37
- setgid* 37
- <setjmp.h>* 39
- setlocale* 37
- setpgid* 37
- setsid* 37
- setuid* 37
- setvbuf* 37
- sigaction* 37
- sigaddset* 37
- sigdelset* 37
- sigemptyset* 37
- sigfillset* 37
- sigismember* 37
- signal* 37
- <signal.h>* 36, 37
- sigpending* 37

sigprocmask 37
sigsuspend 37
slash 11
sleep 37
sleep
 POSIX_CURRENT_PROCESS 37
sprintf 37
srand 38
sscanf 38
start_tag
 POSIX_CGI 28
stat 16, 38
status
 POSIX_FILE_DESCRIPTOR 17, 36
STC_TEMPORARY_FILE 30
<stdarg.h> 39
STDC_BASE ii, 3, 40
STDC_BINARY_FILE 30
STDC_CONSTANTS ii, 3, 30, 41
STDC_CURRENT_PROCESS ii, 30, 42, 97
STDC_DYNAMIC_MEMORY ii, 30, 43, 44
STDC_ENV_VAR ii, 30, 46
STDC_FILE ii, 36, 47, 48, 50, 97
STDC_FILE_SYSTEM ii, 30, 52
STDC_LOCALE_NUMERIC ii, 36, 97
STDC_PATH ii, 97
STDC_SHELL_COMMAND 30, 38
STDC_STATUS ii, 97, 97
STDC_SYSTEM ii, 30, 53
STDC_TEXT_FILE 30
STDC_TIME ii, 30, 35, 54, 97
<stdio.h> 33, 33, 35, 36, 37, 38
<stdioh> 36
<stdlib.h> 35, 36, 37, 38
strftime 38
support
 commercial iv
SUS_SYSLOG iii, 29, 98
synchronize
 POSIX_FILE_DESCRIPTOR 36
synchronize_data
 POSIX_FILE_DESCRIPTOR 36
<sys/staat.h> 37
<sys/stat.h> 35, 36, 37, 38
<sys/utsname.h> 38
<sys/wait.h> 38
sysconf 38
system 38

t
tcdrain 38
tcflow 38
tcflush 38
tcgetattr 38
tcgetpgrp 38
tcsendbreak 38
tcsetattr 38
tcsetpgrp 38
tell
 STDC_FILE 36
temporary_file_name
 STDC_FILE_SYSTEM 38
terminal 13
<termios.h> 35
time 38
<time.h> 35, 36, 37, 38
times 38
<times.h> 38
tmpfile 38
tmpnam 38
to_local
 STDC_TIME 36
to_utc
 STDC_TIME 36
touch
 POSIX_FORK_ROOT 38
ttyname 38
ttyname
 POSIX_FILE_DESCRIPTOR 38
tzset 38

u
umask 38
uname 38
ungetc 38
ungetc
 STDC_FILE 38
<unistd.h> 35, 36, 37, 38
unlink 2, 38
unlink
 POSIX_FILE_SYSTEM 38
utime 38

utime

POSIX_FILE_SYSTEM 38

<utime.h> 38

v*value*

STDC_ENV_VAR 36

vfprintf 38

vprintf 38

vsprint 38

w

wait 38

wait

POSIX_FORK_ROOT 5

POSIX_CURRENT_PROCESS 5, 38

wait_for

POSIX_CHILD 5

wait_pid

POSIX_FORK_ROOT 38

waited_child_pid

POSIX_FORK_ROOT 5

waitpid 38

wcstombs 38

wctomb 38

write 38

write

POSIX_FILE_DESCRIPTOR 38

STDC_FILE 36

x

XML_GENERATOR ii, 95, 96