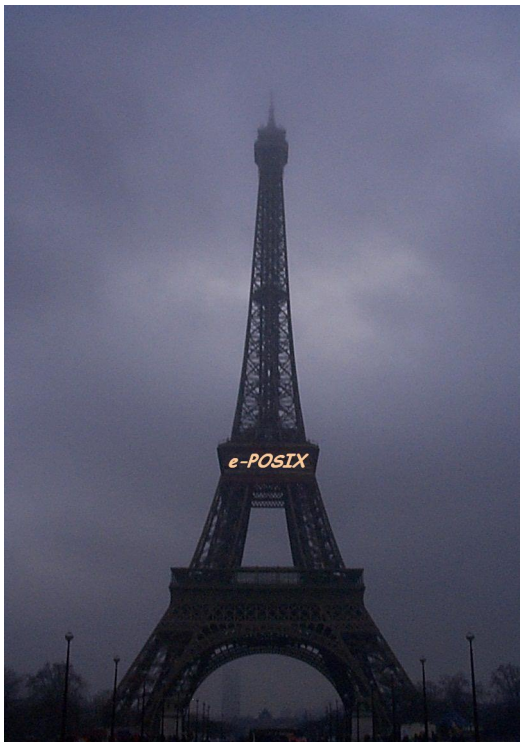

e-POSIX

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

written by Berend de Boer

Contents

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 particularly this library. Prices are 1000 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.
- **mico/E team**: I got many ideas for my C interface from the mico/E project. Sometime ago **Andreas Schulz** wrote me that the micoe team wanted to use e-POSIX in mico/E. Andreas also reported problems and suggested improvements, especially in the *EPX_CGI* class.
- **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.
- **Marcio Marchini <mqm@magma.ca>** contributed a lot to e-POSIX. He gave very useful advice, submitted code, and supplied patches to compile e-POSIX better on Windows. I think it is fair to say that you thank the Windows support in e-POSIX to Marcio.
- **Eric Bezault**: I've had some interesting discussions with Eric regarding architecture of a library as e-POSIX. I think we never agreed :-), but the alternative error handling is due to his comments!

Colophon

The text of this manual was entered with GNU Emacs 20.5.1 on RedHat Linux 6.2. It was typeset with pdfTEX using the ConTEXt macro package, see `http://www.pragma-ade.com`. BON diagrams were created with METAPOST.

In this chapter:

1 Installation

1.1 Compiling the C code

To use e-POSIX, a few C files need to be compiled into a library. This library is called `libeposix.a` on Unix systems and `libeposix.lib` on Microsoft Windows systems.

1.1.1 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 `libeposix.a` before you can use the e-POSIX classes.

This can be done with:

```
make libeposix.a
```

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

1.1.2 Compiling on Windows

For Windows system, I've supplied a tool —build with e-POSIX— that can build the necessary e-POSIX library for your Eiffel and C compiler.

Type:

```
makelib
```

to get help. Type:

```
makelib -ise -msc
```

to compile the C code with Microsoft's Visual C compiler targeting the ISE Eiffel compiler.

Only the Microsoft supplied library did work, i.e. link, with VisualEiffel:

```
makelib -ve -msc
```

Type:

```
makelib -se -bcc
```

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

Type:

```
makelib -se -lcc
```

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

If you have both the Borland C compiler and lcc installed, make sure the make.exe in your path is the correct one!

1.2 Vendor specific notes

1.2.1 ISE Eiffel

The only supported ISE Eiffel compiler is 5.1. This is due to the fact e-POSIX is written to use ELKS2001. ISE Eiffel 4.5, released in 1999, is not ELKS2001 compliant, but see below.

I've tested ISE Eiffel 5.1 on the following conditions:

1. I used Microsoft Windows 2000, Service Pack 2.
2. I used the Borland C 5.5 compiler.

A typical `Ace.ace` file is shown below. It is Windows specific:

```
system
      "eposix_test"

root
      TEST_EPX_ALL: make

default
      assertion (check)
      assertion (require)
      assertion (ensure)
      assertion (loop)
      assertion (invariant)
      arguments (" ")
      disabled_debug (yes)
      debug (no)
      line_generation (no)
      profile (no)
      trace (no)
      working_directory ("m:\tmp")
      il_verifiable (yes)
      msil_generation_type ("exe")
      check_vape (yes)
      console_application (yes)
      address_expression (no)
      array_optimization (no)
      dead_code_removal (yes)
      dynamic_runtime (no)
      exception_trace (no)
      inlining (no)
      multithreaded (no)

-- Ace file for eposix_test
-- Suited for Eiffel 5.1 and Borland C 5.5
```

```
cluster
      root_cluster:         "$EPOSIX\test_suite"

      support_cluster:              "$EPOSIX\src\support"

      capi_cluster:         "$EPOSIX\src\capi"

      stdc_cluster:         "$EPOSIX\src\standardc"
            visible
                  stdc_signal_switch
                        export
                              "switcher"
                        end
            end

      abstract_cluster:             "$EPOSIX\src\abstract"

      wapi_cluster:         "$EPOSIX\src\wapi"

      windows_cluster:              "$EPOSIX\src\windows"

      epxc_cluster:         "$EPOSIX\src\epxc"

      compatibiliy_cluster:          "$EPOSIX\src\spec\ise"

      library base:         "$ISE_EIFFEL/library/base"
            exclude
                  "table_eiffel3"; "desc";
            end

external

      object:
            "$EPOSIX\lib\libeposix-ISE-BCC.lib"

end
```

This file is also provided in the `test_suite` directory.

Previous versions of e-POSIX worked with ISE Eiffel 4.5 by following the steps documented below. Because 4.5 is no longer supported, I do not test it against 4.5 anymore. But I will attempt to repair reported failures of this procedure:

1. Make sure their is an environment variable `EPOSIX`, pointing to the root of the e-POSIX hierarchy.
2. The *STRING.is_empty* needs to be replaced by *empty*. If you have a Unix like shell, you can run the `build_ise.sh` which does this automatically. A quite complete Unix environment is provided by Cygwin. You can download it from `http://www.redhat.com`.

1.2.2 SmallEiffel

e-POSIX was developed using SmallEiffel -0.74beta12 on FreeBSD and Linux.

To successfully compile with SmallEiffel you need:

1. The environment variable `EPOSIX`, which should be set to the directory where you have installed e-POSIX.
2. A correct `libeposix.a` or `libeposix.lib`, see **section 1.1**. If you have the default eljwin32 installation, it is as easy as:
   ```
   makelib -lcc -se
   ```
   The library is placed in the subdirectory `lib`.
3. A correct `loadpath.se` or Ace file. Default `loadpath.se`'s have been supplied in in various directories, depending on if you want to write Standard C, Windows or POSIX compatible programs. A Standard C specific `loadpath.se` is in the `src/windows` directory, a POSIX specific one is in `src/posix`.
4. Pass either the `libeposix.a` library or object files to the compiler.

On Unix, compiling a class which uses e-POSIX can be done with:

```
compile MYCLASS make  -L$EPOSIX/lib -leposix
```

Make sure the `-L` option points to the directory `libeposix.a` can be found. Usually this is `eposix/lib`. In case signal handling is included —which is probably included earlier than you expect— you might have to compile with a cecil file. With this file, callbacks from Unix to Eiffel are possible:

```
compile MYCLASS make -L$EPOSIX/lib -leposix -cecil src/supportc/cecil.se
```

On Windows, compiling a class which uses e-POSIX can be done with:

```
compile -no_style_warning test_all make $(EPOSIX)\lib\libeposix.lib
```

Make sure that the last argument includes the directory where `libeposix.lib` resides. This command–line seems to work for all three supported compilers.

If callbacks are necesary, because something is done with signals, use the following command:

```
compile \
  -cecil src\supportc\cecil.se
  -no_style_warning test_all make $(EPOSIX)\lib\libeposix.lib
```

Note that the backslashes are continuation characters, everything should be typed on a single line. This line is just broken up for typesetting purposes.

On POSIX systems, a typical SmallEiffel `loadpath.se` looks like:

```
${EPOSIX}/src/epxp/
${EPOSIX}/src/posix/
${EPOSIX}/src/papi/
${EPOSIX}/src/abstract/
${EPOSIX}/src/epxc/
${EPOSIX}/src/standardc/
${EPOSIX}/src/capi/
${EPOSIX}/src/spec/se/
${EPOSIX}/src/support/
```

On Windows, where most of the POSIX API is not available, `loadpath.se` can look like:

```
${EPOSIX}/src/windows/
${EPOSIX}/src/wapi/
${EPOSIX}/src/abstract/
${EPOSIX}/src/epxc/
${EPOSIX}/src/standardc/
${EPOSIX}/src/capi/
${EPOSIX}/src/spec/se/
${EPOSIX}/src/support/
```

Note that you can easily use the supplied `loadpath.se`'s by writing your own `loadpath.se` which contains just:

```
${EPOSIX}/src/posix/loadpath.se
```

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.

The latest SmallEiffel offers support for Ace files. A typical Ace file for e-POSIX looks like this:

```
system

  eposix_test

root

  TEST_ALL: make

default

  no_style_warning (yes)
  collect (no)
  assertion (all)
  debug (yes)
  trace (no)

cluster

  current: "."

  eposix_testsuite: "${EPOSIX}/test_suite"

  eposix_support: "${EPOSIX}/src/support"
  eposix_spec: "${EPOSIX}/src/spec/se"
  eposix_capi: "${EPOSIX}/src/capi"
  eposix_standardc: "${EPOSIX}/src/standardc"
  eposix_epxc: "${EPOSIX}/src/epxc"
```

```
  se_io: "${SmallEiffelDirectory}lib/io"
    default
      assertion (require)
    end

  se_kernel: "${SmallEiffelDirectory}lib/kernel"
    default
      assertion (require)
    end

  se_base: "${SmallEiffelDirectory}lib/base"
    default
      assertion (require)
    end

external

  external_lib_path: "-L${EPOSIX}/lib"

  external_lib: "-leposix-SE"

  cecil ("${EPOSIX}/src/supportc/cecil.se")

generate

  no_split (yes)
  c_compiler_options: "-pipe -O0"

end
```

1.2.3 Visual Eiffel

e-POSIX has been tested with ObjectTool's free VisualEiffel 4 for Linux. e-POSIX should work out of the box. You have to make the e-POSIX library as explained in **section 1.1**. In your projects you can use the eposix cluster, see `src/cluster.es`.

Follow these steps to compile with VisualEiffel 4 on Windows:

1. Make sure the VE_BIN environment variable is set to the Bin directory in the VisualEiffel subdirectory. On my system it is set to `M:/Program Files/ObjectTools/VisualEiffel/Bin`.
2. Create the `libeposix.lib` library using the Microsoft Visual C compiler:
   ```
   makelib -ve -msc
   ```
3. Add a cluster with name eposix, pointing to the `src` directory. The provided `src/cluster.es` file will give you a correct cluster. The provided `cluster.es` should work both on Windows and Unix.
4. Create a new project. Make sure to set the linker supplier option to Microsoft!

Versions of VisualEiffel prior to 4 are not supported. However, previous versions of e-POSIX compiled almost out of the box with Visual Eiffel 3.3 beta, which was ELKS2000 compliant. My beta missed `cecil.h` which was sent to me separately by ObjectTools. If you miss it, you might want to contact them.

Because VisualEiffel 3.3 beta does not support the **create** keyword, use the provided `build_ve.sh` script to replace all **create**s by the bang bang syntax.

This is a Unix shell script, you there for need a Unix shell on NT. A quite complete Unix environment is provided by Cygwin. You can download it from `http://www.redhat.com`.

1.2.4 Halstenbach Eiffel

e-POSIX has not been tested with this compiler.

1.3 Platform specific notes

Although e-POSIX should, in principle, run on every platform that supports Standard C or POSIX, it cannot be tested on every platform by me alone. This section gives details about the platforms I've used.

1.3.1 Linux

The latest version of e-POSIX was tested with RedHat 7.1 with kernel 2.4.17.

1.3.2 FreeBSD

The latest version of e-POSIX was tested with FreeBSD 4.4-STABLE. FreeBSD doesn't support `fdatasync`, so we do a `fsync`there. With release 1.1 I expect to have converted to use the configure script to automatically detect such cases.

1.3.3 Cygwin

The latest version of e-POSIX was tested with Cygwin 1.3.x. Some remarks:

1. Locking doesn't seem to be supported.
2. fifo's (`mkfifo`) are not supported.
3. No support for `fdatasync`, so we do a `fsync`there.

1.3.4 BeOS

The latest version of e-POSIX was tested with BeOS 5.03. BeOS has a nice POSIX compatibility layer. Some remarks:

1. Locking doesn't seem to be supported.
2. fifo's (`mkfifo`) are not supported.

3. Hard links are not supported, only symbolic links.
4. No support for `fdatasync`, so we do a `fsync`there.

1.3.5 QNX

The latest version of e-POSIX was tested with QNX ?.

1.3.6 Win32

The latest version of e-POSIX was tested with Windows 2000, Service Pack 2. On Win32 Standard C is fully supported. With e-POSIX's abstract layer, parts of POSIX and the Single Unix Specification are also supported. Support isn't as extensive as using the Cygwin tools.

In this chapter:

2 Design notes

2.1 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 nor 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. For example they usually have much logic at the C level. I wanted only C glue code: all intelligence should be in the Eiffel code.

Another attempt is done by the Gobo cluster: it attempts to provide users with a set of classes that work accross 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.

2.2 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. Uuse 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.

2.3 Class structure

e-POSIX makes available all the Standard C and POSIX headers in classes like *CAPI_STDIO* and *PAPI_UNISTD*. You can find more details about the header translation in **chapter 11**.

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, while 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 2.1**.

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 2.2**:

1. If a class builds upon facilities available on Standard C, its name starts with the prefix STDC_ and it inherits from *STDC_BASE*.
2. If a class builds upon facilities available in POSIX, its name starts with the prefix POSIX_ and it inherits from *POSIX_BASE*.
3. If a class builds upon facilities available in the Single Unix Specification, its name starts with the prefix SUS_ and it inherits from *SUS_BASE*. The support for the Single Unix Specification is not yet complete, but is continually enhanced.
4. Because we live in a world dominated by Microsoft Windows, and Microsoft Windows does not do POSIX, this would mean that many users only could use e-POSIX's Standard C facilities. These facilities are extremely limiting, for example there is no change directory command in Standard C. Therefore e-POSIX makes available an abstraction layer that covers routines that have an equivalent in POSIX and the Single Unix Specification. These classes start with the name EPX_. They always inherit from classes starting with ABSTRACT_. These abstract classes implement the common code. See **chapter 8.2** for more details.

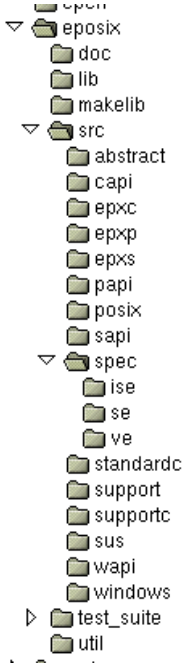

Figure 2.1 e-POSIX directory structure

Note that by using Cygwin you have a full POSIX emulation layer on Windows. In that specific environment you can use e-POSIX's entire POSIX and Single Unix Specification layer.

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.

2.4 Clients of this library

For client classes, two important classes are *STDC_CONSTANTS* and *POSIX_CONSTANTS*, see **figure 2.3**. 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.

2.5 Forking

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

class *PROCESS*

feature

pid: *INTEGER*

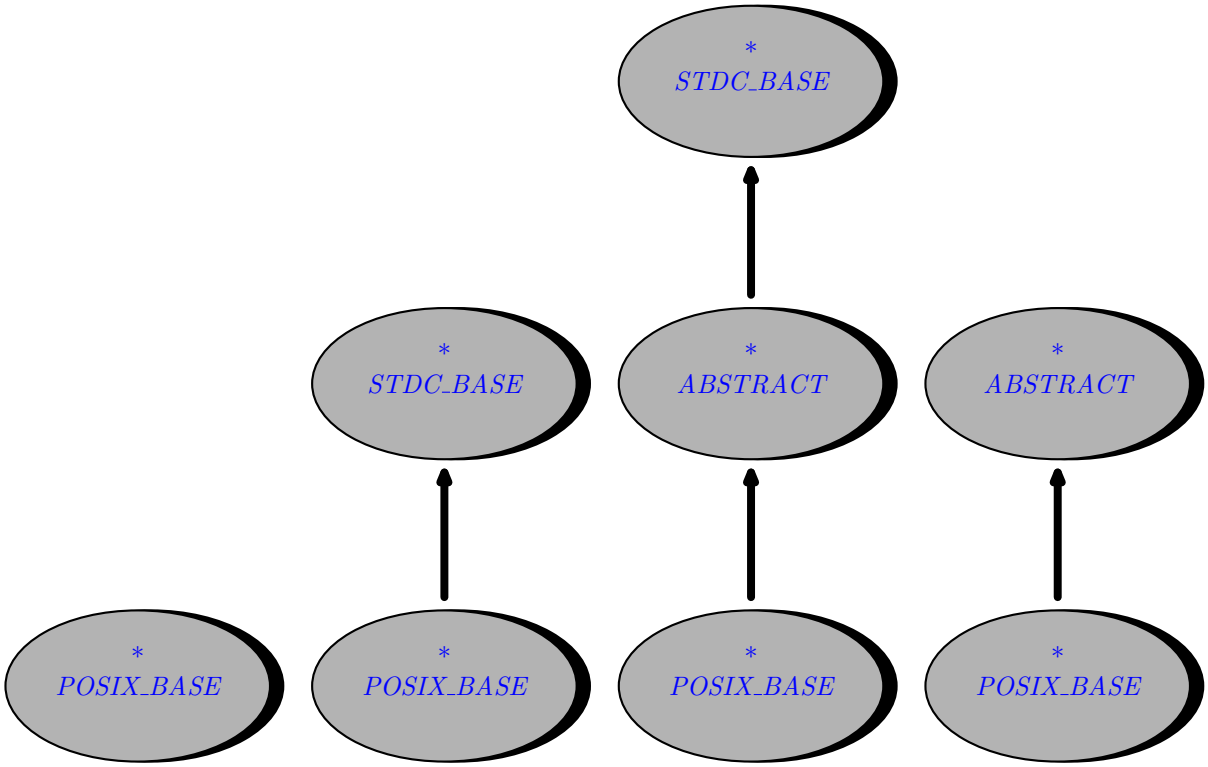

Figure 2.2 Inheritance structure

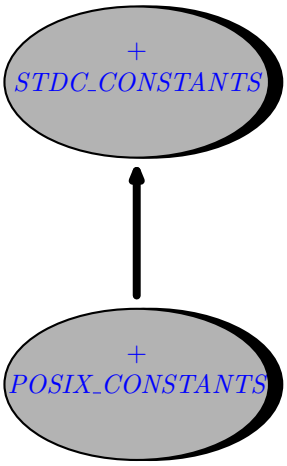

Figure 2.3 Standard C and POSIX constants

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)
            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.

2.6 Books

Books that have been helpful during the development of e-POSIX where [1–3], see the biography section at **page 194**.

In this chapter:

- ***Working with files***
- ***Working with file descriptors***
- ***Working with the file system***
- ***Executing a child command***
- ***Current time***
- ***Accessing environment variables***
- ***Allocating memory***
- ***Redirecting stderr to stdout***

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 6**.

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*. According to the POSIX standard, there is no distinction between binary and text files. But on certain systems you must use POSIX programs through an emulation layer. For example, on Windows Cygwin is a well–known POSIX emulator. To maintain compatibility with other Windows programs, Cygwin distinguishes between text and binary files. If you use Cygwin to compile your POSIX programs, this distinction is therefore still important.

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
                file.read_string (256)
            until
                file.eof
            loop
```

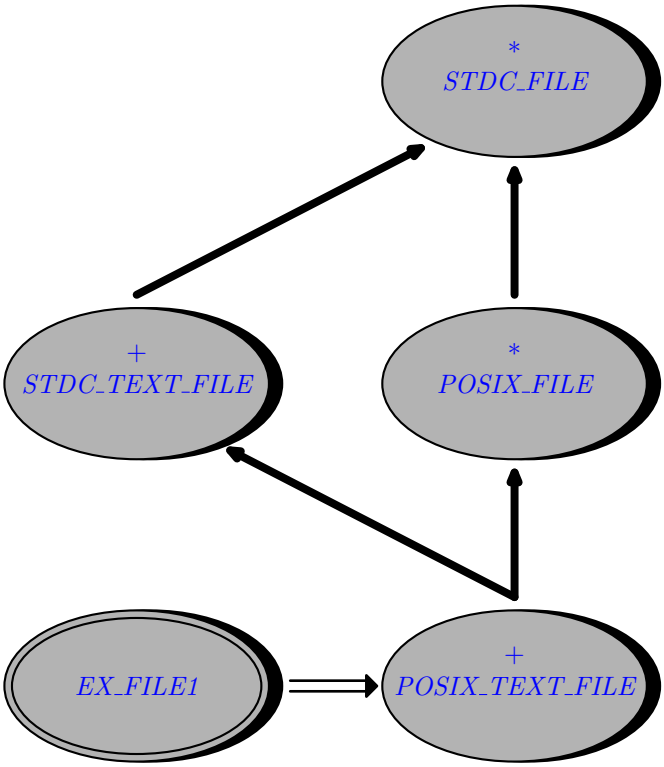

Figure 3.1 BON diagram of opening a text file.

```
        print (file.last_string)
        file.read_string (256)
    end
    file.close
end

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.

Every line that is read includes the end–of–line character if one was present. This is unlike Pascal for example, but more like Perl. e-POSIX provides the feature *POSIX_TEXT_FILE.chop* which removes the last character of *last_string* if and only if it is an end–of–line character. And that is unlike Perl, which removes any character. With e-POSIX it is not necesary to test for the end–of–line characters if you just want to remove it in case one is present.

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 garbaged 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.

Reading binary files is almost the same loop, only you read it in chunks:

```
class EX_FILE2

creation

    make

feature

    chunk_size: INTEGER is 512

    make is
        local
            file: POSIX_BINARY_FILE
            buffer: POSIX_BUFFER
        do
            create file.open_read ("/bin/sh")
            create buffer.allocate (chunk_size)
            from
                file.read_buffer (buffer, 0, chunk_size)
            until
                file.eof
            loop
                file.read_buffer (buffer, 0, chunk_size)
            end
            file.close
        end

end
```

This example uses a more safe version of buffer reading, *POSIX_FILE.read_buffer*. There is an untyped variant *POSIX_FILE.read* which accepts a pure pointer. There is no need to mention that you need to watch buffer overflows carefully with this last one!

Correctly looping through files, takes care. For example the following loop also works, but is less correct:

```
class EX_WRONG1

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

end
```

After *POSIX_TEXT_FILE.read_string*, *eof* might be True. Because the string is empty in that case, nothing will be printed. You will make an unnecessary extra loop. The correctly coded variant is:

```
class EX_WRONG2

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)
                if not file.eof then
                    print (file.last_string)
                end
            end
            file.close
        end

end
```

I myself prefer the first example, as the check is only in the **until** part, and not repeated in the loop.

The following examples shows how a binary file is created and a string is written to it.

```
class EX_FILE3

inherit

    POSIX_FILE_SYSTEM
```

```
creation

    make

feature

    make is
        local
            file: POSIX_BINARY_FILE
        do
            create file.create_write (expand_path ("$HOME/myfile.tmp"))
            file.write_string ("hello world.%N")
            file.close
        end

end
```

Depending on the platform you are running a backslash is turned into a slash or vice versa.

This example also demonstrates how path names —file and directory names— can be expanded: if you call *POSIX_FILE_SYSTEM.expand_path*, any environment variables in the path are expanded. Backslashes and slashes are always translated, but environment variable expansion has to be done explicitly.

You can move the file pointer with two different methods: *POSIX_FILE.seek* and *set_position*. The *seek* works with files up to 2 GB, *set_position* has no such limits. Use *tell* to get a position that can be passed to *seek*. Use *get_position* to get a position that can be passed to *set_position*.

```
class EX_FILE5

creation

    make

feature

    make is
        local
            file: POSIX_BINARY_FILE
            pos1: INTEGER
            pos2: STDC_FILE_POSITION
        do
            create file.create_read_write ("test.bin")
            file.write_string ("one")
            pos1 := file.tell
            pos2 := file.get_position
            file.write_string ("two")
            file.seek (pos1)
            -- or file.set_position (pos2)
            file.read_string (3)
```

```
            if not file.last_string.is_equal ("two") then
                print ("unexpected read.%N")
            end
            file.close
        end

end
```

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
```

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
                file.read_string (256)
            until
                file.eof
            loop
                print (file.last_string)
                file.read_string (256)
            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
                some_lock,
                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)
                some_lock := fd.get_lock (lock)
                if some_lock /= Void then
                    print ("There is already a lock?%N")
                end

                -- create exclusive lock
                lock.set_allow_none
                lock.set_start (0)
```

```
            lock.set_length (4)
            fd.set_lock (lock)

            fd.close
        end

    end
```

POSIX_FILE_DESCRIPTOR.get_lock is command–query separated, that is why it returns a new lock when queried and there is a lock. If there is no lock *get_lock* returns Void. The passed lock is not modified.

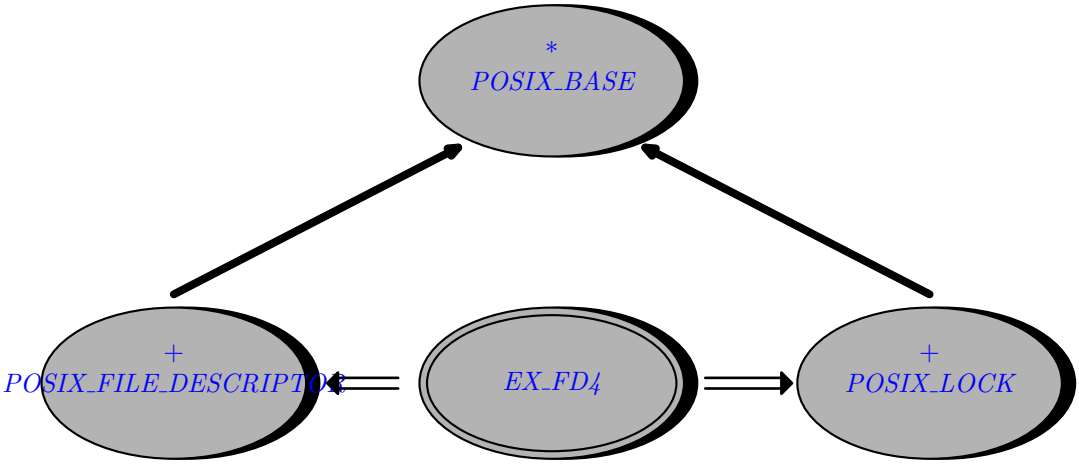

Figure 3.2 BON diagram of locking a portion of a file.

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

            -- 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

creation

    make

feature

    make is
        do
            change_directory (expand_path ("~"))
            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.

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

creation

    make

feature

    make is
        local
            perm: POSIX_PERMISSIONS
        do
            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

    print_info (ok: BOOLEAN; what: STRING) is
        do
            print ("is_")
            print (what)
            print (" returned ")
            print (ok)
            print (".%N")
        end
```

```
end
```

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

As can be seen in the above example, one 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

creation

    make

feature

    make is
        local
            stat: POSIX_STATUS
        do
            stat := status ("/etc/passwd")
            print ("size: ")
            print (stat.size.out)
            print (".%N")
            print ("uid: ")
            print (stat.permissions.uid)
            print (".%N")
        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
```

```
creation

    make

feature

    make is
        local
            dir: POSIX_DIRECTORY
        do
            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.

When browsing a directory, all entries in that directory are returned. You might want to be interested only in certain files. e-POSIX has the ability to define arbitrary filters. Standard e-POSIX comes with an extension filter that only shows files with a certain extension:

```
class EX_DIR6

inherit

    POSIX_FILE_SYSTEM

creation

    make

feature

    make is
        local
            dir: POSIX_DIRECTORY
        do
            from
                dir := browse_directory (".")
```

```
            dir.set_extension_filter (".e")
            dir.start
        until
            dir.exhausted
        loop
            print (dir.item)
            print ("%N")
            dir.forth
        end
        dir.close
    end

end
```

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
```

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_EXEC1

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", <<"-1", ".">>)
            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.

It is important to wait for the child that has been executed at some point in time, just like any POSIX would have to do. If you do not wait for a child process, memory in the kernel is not released and eventually you would run out of processes. Also, after the *POSIX_EXEC_PROCESS.wait_for* command, the exit code of the process becomes available.

3.5 Current time

e-POSIXhas 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. Do this by calling *to_local* or *to_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.

```
class EX_ENV1

creation

    make

feature

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

end
```

Unfortunately, it is not possible in POSIX to set an environment variable. This is possible with the Single Unix Specification classes, see **section 5.1**.

3.7 Allocating memory

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

```
class EX_MEM

creation

    make
```

```
feature

    make is
        local
            mem: POSIX_BUFFER
            byte: INTEGER
        do
            create mem.allocate (256)
            mem.poke_uint8 (2, 57)
            byte := mem.peek_uint8 (2)
            mem.resize (512)
            mem.deallocate
        end

end
```

For more information about the dynamic memory class, see **section 6.1**.

3.8 Redirecting stderr to stdout

If you want to redirect all output written by your program or any child you spawn to stdout, you can use the *POSIX_FILE_DESCRIPTOR.make_as_duplicate* call:

```
class EX_REDIRECT1

inherit

    POSIX_CURRENT_PROCESS

creation

    make

feature

    make is
        do
            -- flush stream buffers, else output may be in wrong order
            stdout.flush
            stderr.flush

            fd_stderr.make_as_duplicate (fd_stdout)
            -- all output written to stderr goes to stdout now
        end

end
```

It's a good idea to call this at the beginning of your program, before you have written anything to stderr or stdout. If you do that, you don't have to flush the stream buffers.

In this chapter:

- ***Catching a signal***
- ***General wait for child handler***
- ***Forking a child process***
- ***Creating a daemon***
- ***Asynchronous I/O***
- ***Talking to your modem***
- ***Using shared memory***
- ***More examples***

4 Advanced Posix examples

4.1 Catching a signal

Every class can become a signal handler by inheriting from *POSIX_SIGNAL_HANDLER*. Implement the *signalled* method as that is the function that is called when the signal occurs. Use *POSIX_SIGNAL.set_handler* to make your class a signal handler and call *apply* to start receiving signals when they occur.

The following examples demonstrates a program that can be safely interrupted by pressing Ctrl+C:

```
class EX_SIGNAL1

inherit

    POSIX_CURRENT_PROCESS

    POSIX_CONSTANTS

    POSIX_SIGNAL_HANDLER

creation

    make

feature

    handled: BOOLEAN

    make is
        local
            signal: POSIX_SIGNAL
        do
            create signal.make (SIGINT)
            signal.set_handler (Current)
            signal.apply
```

```
            print ("Wait 30s or press Ctrl+C.%N")
            sleep (30)
            if handled then
                print ("Ctrl+C pressed.%N")
            else
                print ("Ctrl+C not pressed.%N")
            end
        end

    signalled (signal_value: INTEGER) is
        do
            handled := True
        end

end
```

All precautions and warnings when handling signals in C apply equally well in Eiffel of course.

You can write a single signal handler, that handles multiple signals. This makes it possible to have signal handling code in just one place. Create a class that inherits from *POSIX_SIGNAL_HANDLER*. Pass this class to the *POSIX_SIGNAL.set_handler* for every signal you want to catch. The signal value is passed as parameter to *POSIX_SIGNAL_HANDLER.signalled*, so you can write an **inspect** statement based on the value.

4.2 General wait for child handler

If you do not want to wait for every child process explicitly, you can write a simple SIGCHLD handler that just does a wait (I found this idea in [4]):

```
class EX_SIGNAL2

inherit

    POSIX_CURRENT_PROCESS

    POSIX_CONSTANTS

    POSIX_SIGNAL_HANDLER

creation

    make

feature

    make is
        local
            signal: POSIX_SIGNAL
        do
```

```
            create signal.make (SIGCHLD)
            signal.set_handler (Current)
            signal.apply

            -- spawn child processes here
            -- you dont have to wait for them
        end

    signalled (signal_value: INTEGER) is
        do
            wait
        end

end
```

In Unix 98 you should be able to set the ignore handler for this signal. In pure POSIX systems the behaviour of the ignore handler is unspecified.

4.3 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

creation

    make

feature

    make is
        local
            reader: POSIX_TEXT_FILE
            stop_sign: BOOLEAN
```

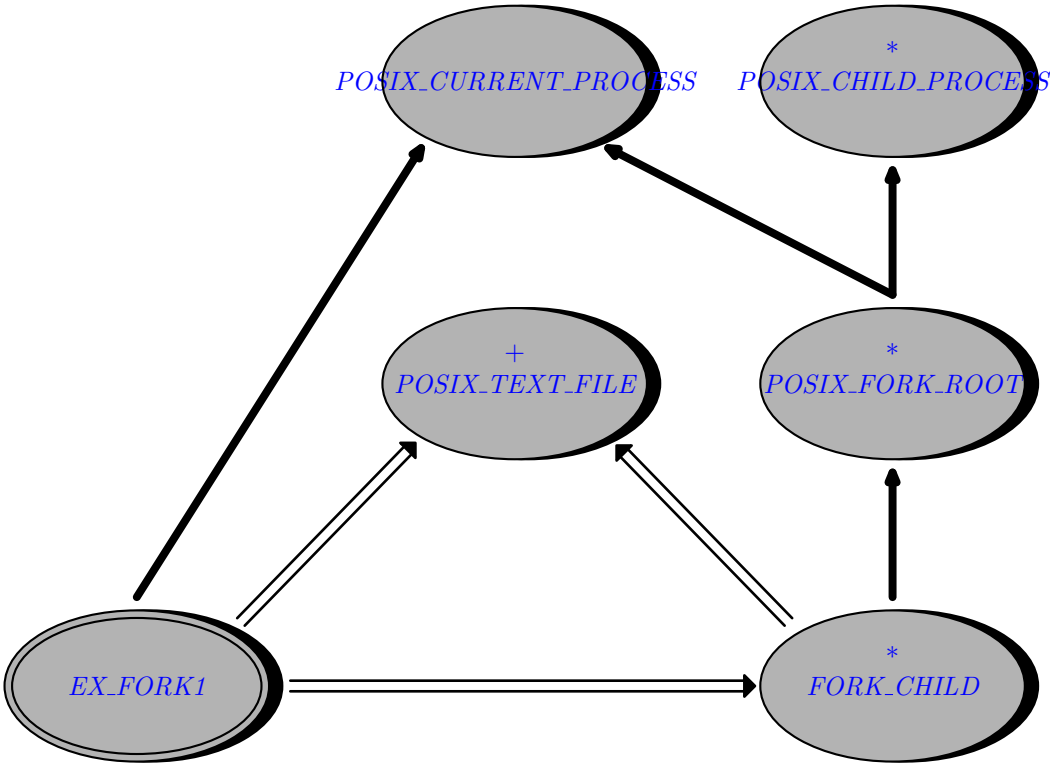

Figure 4.1 BON diagram of forking a child process.

```
    child: FORK_CHILD
do
    -- necessary for SmallEiffel before -0.75 beta 7
    ignore_child_stop_signal

    unlink ("berend.tmp")
    create_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
```

4.4 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
```

4.5 Asynchronous I/O

e-POSIX supports the asynchronous i/o features of POSIX. Not all Free Unices seem to support this feature, nor does their support seems to be error free.

Take a look at the following example:

```
class EX_ASYNC1

creation

    make
```

```
feature

    make is
        local
            fd: POSIX_FILE_DESCRIPTOR
            request: POSIX_ASYNC_IO_REQUEST
        do
            create fd.create_read_write ("test.tmp")
            create request.make (fd)
            request.set_offset (0)
            request.write_string ("hello world.")
            request.wait_for
            fd.close
        end

end
```

The basic idea is that each asynchronous request is a separate object, modeled by *POSIX_ASYNC-_IO_REQUEST*. You prepare it through calls like *set_buffer*, *set_count* and *set_offset*. You execute the request by calling *read* or *write*.

You can wait for the request to be complete by calling *wait_for*. It should be possible to force open requests to be synchronized to the disk with *synchronize*, but this does give strange results on Linux. So far I haven't got access to a machine that also implements asynchronous i/o to test if my code is correct.

4.6 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 ("ATI0%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
```

4.7 Using shared memory

You can use shared memory to exchange data between different processes. It's dependent on your POSIX version if this is supported, so check for this capability explicitly!

```
class EX_SHARED_MEM1
```

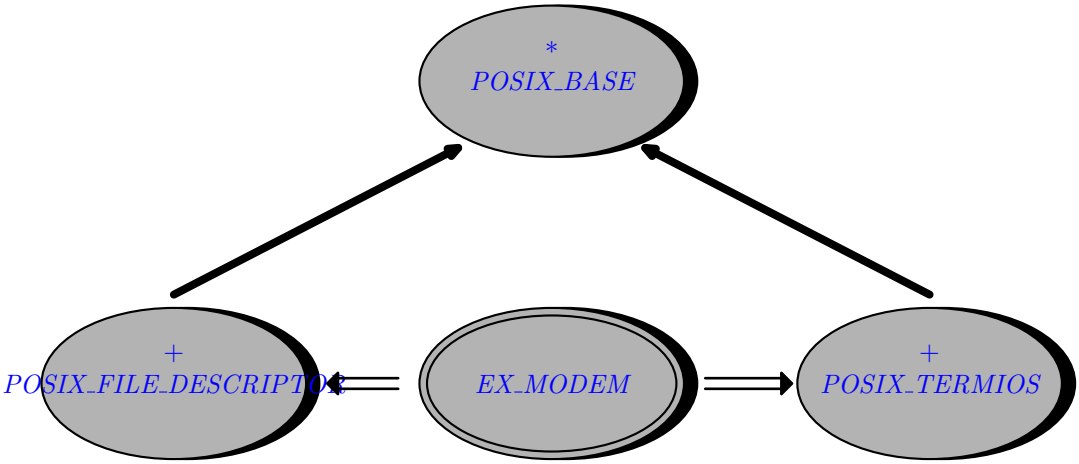

Figure 4.2 BON diagram of talking to a modem.

```
inherit

    POSIX_SYSTEM

    POSIX_CURRENT_PROCESS

    POSIX_FILE_SYSTEM

creation

    make

feature

    make is
        local
            fd: POSIX_SHARED_MEMORY
        do
            if not supports_shared_memory_objects then
                stderr.puts ("Shared memory objects not supported.%N")
                exit_with_failure
            end

            create fd.create_read_write ("/test.berend")
            fd.write_string ("Hello world.%N")
            fd.close
            unlink_shared_memory_object ("/test.berend")
        end

end
```

Make sure you always start a shared memory object with a slash. Else the behaviour is undefined or processes might not be able to find your shared memory.

4.8 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.

In this chapter:

- ***Environment variables***
- ***Logging messages and errors***
- ***Sockets***

5 Single Unix Specification classes

5.1 Environment variables

Using *SUS_ENV_VAR*set_value it is possible to set environment variables.

5.2 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_CONSTANTS

    SUS_SYSLOG_ACCESSOR

creation

    make

feature

    make is
        do
            syslog.open ("test", LOG_ODELAY + LOG_PID, LOG_USER)

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

            syslog.close
        end
```

```
end
```

Always use the *SUS_SYSLOG_ACCESSOR* to access the syslog wrapper class *SUS_SYSLOG*. *SUS_SYSLOG* is a singleton, it makes no sensen to open a connection to the syslog daemon twice.

5.3 Sockets

e-POSIX currently has initial socket support. It will work only on POSIX systems, support for Windows through the EPX layer will be added in a future release.

The following example demonstrates a simple echo client:

```
class EX_ECHO1

creation

    make

feature

    hello: STRING is "Hello World.%N"

    make is
        local
            host: SUS_HOST
            service: SUS_SERVICE
            echo: SUS_TCP_SOCKET
            sa: SUS_SOCKET_ADDRESS
        do
            -- create host.make_from_name ("localhost")
            create host.make_from_name ("bmach")
            create service.make_from_name ("echo", "tcp")

            create sa.make (host, service)

            create echo.open_by_address (sa)
            echo.write_string (hello)
            echo.read_string (256)
            if not echo.last_string.is_equal (hello) then
                print ("!! got: ")
                print (echo.last_string)
            end
        end

end
```

In this chapter:

- ***Allocating memory***
- ***Accessing environment variables***
- ***Working with streams***
- ***Working with the file system***

6 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_BUFFER*: 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.

6.1 Allocating memory

You can dynamically allocate memory with *STDC_BUFFER* which works just like *POSIX_BUFFER*.

```
class EX_MEM2

creation

    make

feature

    make is
        local
            mem: STDC_BUFFER
            byte: INTEGER
        do
            create mem.allocate_and_clear (128)
```

```
            mem.poke_uint8 (2, 57)
            byte := mem.peek_uint8 (2)
            mem.resize (256)
            mem.deallocate
        end

end
```

With the feature *STDC_BUFFER.allocate_and_clear* memory is allocated and cleared to all zeros.

STDC_BUFFER contains many routines to read bytes and strings from the memory it manages like *peek_int16*, *peek_uint16*, or *peek_int32*. It supports reading and writing 16 and 32 bit integers in little and big endian order with routines as *peek_int16_big_endian*, *peek_int16_little_endian*, and *poke_int32_big_endian*.

6.2 Accessing environment variables

Standard C supports reading environment variables with *STDC_ENV_VAR*.

```
class EX_ENV2

creation

    make

feature

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

end
```

6.3 Working with streams

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

```
class EX_FILE4

creation

    make

feature
```

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

end
```

Its BON diagram, see **figure 6.1** is therefore quite equal to the POSIX one, see **figure 3.1**.

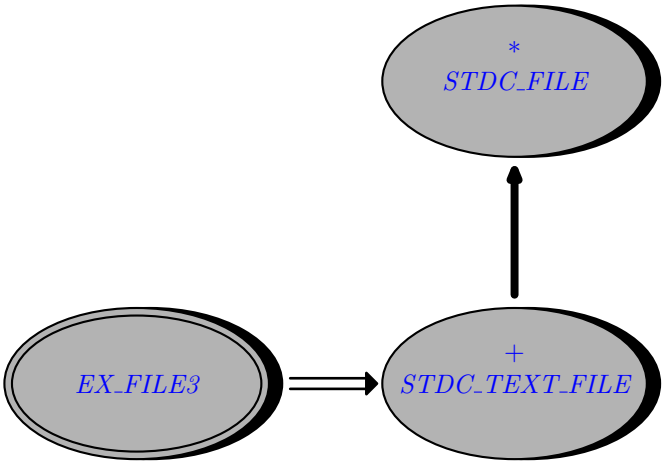

Figure 6.1 BON diagram of opening a Standard C text file.

6.4 Working with the file system

Standard C doesn't offer much for file systems. You can only delete and rename files.

```
class EX_DIR5

inherit

    STDC_FILE_SYSTEM

creation
```

```
    make

feature

    make is
        do
            rename_to ("qqtest.abc.tmp", "qqtest.xyz.tmp")
            remove_file ("qqtest.xyz.tmp")
        end

end
```

The BON diagram is shown in **figure 6.2**.

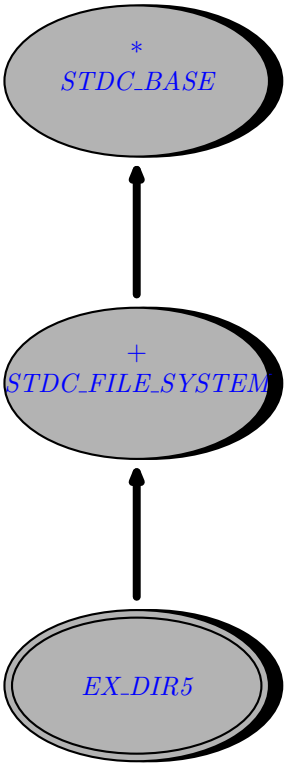

Figure 6.2 BON diagram of deleting and renaming files with Standard C.

In this chapter:

7 Writing CGI programs

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

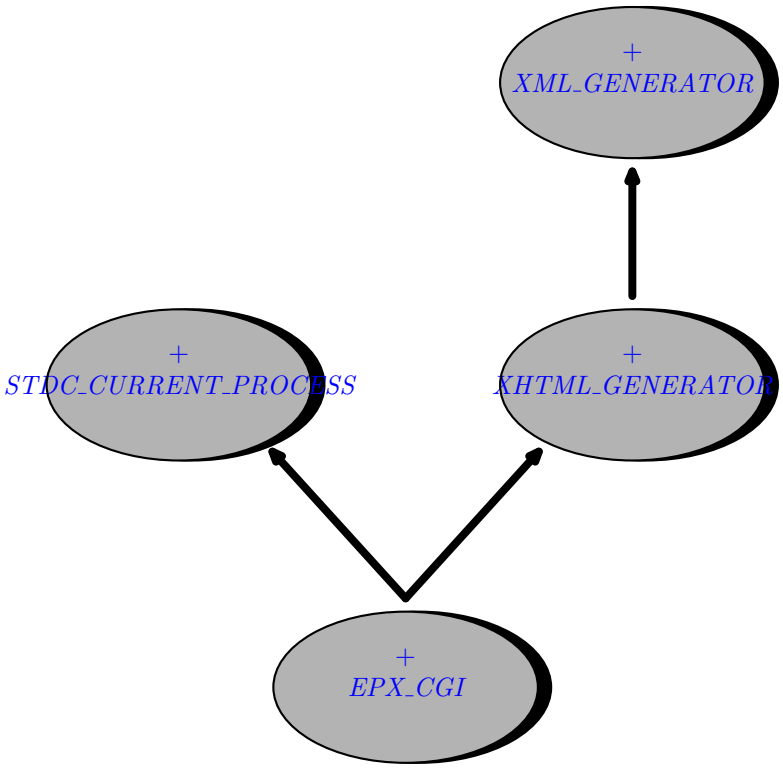

Figure 7.1 BON diagram of *EPX_CGI*.

You just inherit from *EPX_CGI* and start calling its features.

```
class EX_CGI1

inherit

    EPX_CGI

creation
```

```
    make

feature

    execute is
        do
            content_text_html

            doctype
            b_html

            b_head
            title ("e-POSIX CGI example.")
            e_head

            b_body

            p ("Hello World.")
            extend ("<p>you can use your <b>own</b> tags.</p>")
            b_p
            puts ("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
```

Output is written to stdout, but also made available in *my_xml*. Generated output is XHTML, which usually displays fine with older browsers. If strict XHTML is problematic, you can call *doctype_transitional* instead of *doctype*.

It is important not to mix writing to stdout with the features you inherit from *EPX_CGI*. *EPX_CGI* does some caching, so after a tag is started by *EPX_CGI.start_tag* it is not yet written to standard

output. If you want to write something to standard output, use the *EPX_CGI.add_data* feature or its shortcut alias *puts*. If you want to write real tags, use *extend*. This last feature allows you to write anything, while *puts* escapes reserved characters like '>'.

If you use provided features like *b_a*, *b_p* and such, an attempt is made to produce good looking source. Also your input is somewhat validated against XHTML standards.

It is also easy to write a CGI program that displays a form and accepts submitted values. Even file upload is supported. The following example uses the GET method to submit data:

```
class EX_CGI2

inherit

    EPX_CGI

creation

    make

feature

    execute is
        do
            content_text_html

            doctype
            b_html

            b_head
            title ("e-POSIX CGI form example.")
            e_head

            b_body

            b_form_get ("ex_cgi2.bin")

            b_p
            puts ("Name: ")
            b_input ("text", "name")
            set_attribute ("size", "32")
            e_input
            e_p

            b_p
            puts ("City: ")
            input_text ("city", 40, "enter city here")
            e_p
```

```
            b_p
            b_button_submit ("action", "GO!")
            e_button_submit

            nbsp

            button_reset
            e_p

            e_form

            hr

            p ("In your last submit you entered:")
            b_p
            if not exist_value ("name") then
                puts ("!!!!!")
            end
            puts ("name: ")
            puts (value ("name"))
            puts (", ")
            puts ("city: ")
            puts (raw_value ("city"))
            e_p

            e_body
            e_html

        end

    end
```

You can use *EPX_CGI.b_input* to start an input element as shown for the input of a name. Or you can use *input_text* to start a simple text input as shown for the input of a city. Below the line you see the value a user has submitted, if any. Use *value* to get values with certain meta–characters removed. The output is still not save to be passed straight to a Unix Shell though! You can use *raw_value* to get the contents as submitted by the user.

In the above example it doesn't matter much if you use *b_form_get* or *b_form_post*. But with the GET method, you cannot upload files. The following example demonstrates how files can be uploaded:

```
    class EX_CGI3

    inherit

        EPX_CGI

    creation
```

```
    make

feature

    execute is
        do
            content_text_html

            assert_key_value_pairs_created
            save_uploaded_files

            doctype
            b_html

            b_head
            title ("e-POSIX CGI file upload example.")
            e_head

            b_body

            b_form ("post", "ex_cgi3.bin")
            set_attribute ("enctype", multipart_encoding)

            b_p
            puts ("Filename: ")
            b_input ("file", "filename")
            set_attribute ("size", "32")
            set_attribute ("maxlength", "128")
            e_input
            e_p

            b_p
            b_button_submit ("action", "Upload file(s)")
            e_button_submit

            nbsp

            button_reset
            e_p

            e_form

            e_body
            e_html

        end
```

```
    save_uploaded_files is
        local
            i: INTEGER
            kv: EPX_KEYVALUE
            buffer: STDC_BUFFER
            target_name: STRING
            target: STDC_BINARY_FILE
        do
            create buffer.allocate (8192)
            from
                i := cgi_data.lower
            until
                i > cgi_data.upper
            loop
                kv := cgi_data.item (i)
                if kv.file /= Void then
                    from
                        target_name := "/tmp/" + kv.value
                        create target.create_write (target_name)
                        kv.file.read_buffer (buffer, 0, 8192)
                    until
                        kv.file.eof
                    loop
                        target.write_buffer (buffer, 0, kv.file.last_read)
                        kv.file.read_buffer (buffer, 0, 8192)
                    end
                    target.close
                    kv.file.close
                end
                i := i + 1
            end
            buffer.deallocate
        end

end
```

It is important to set the encoding type. This example accepts a file and writes it to `/tmp`. Because multiple files can be present, this example just loops over all key value pairs and checks if a file is present. This example isn't fool–proof with multiple users submitting the same file, but you should get the idea.

Note that the first line is *EPX_CGI.content_text*: in case an exception occurs, the web server is still able to output something back to the user.

After that we make sure that the key value pairs are created with *assert_key_value_pairs_created*. They are automatically created if you call *value*, but in this case we want the key value pairs themselves. In *EX_CGI3.save_uploaded_files* we use the *EPX_KEYVALUE.file* feature to check if that key value pair is an uploaded file: if it is not Void, it points to a temporary file. As this file will be deleted when it is closed or when your program exits, we have to copy it to a new file.

The filename is just the value part of this key value pair. The filename is guaranteed to be free of directory parts.

In the last example we just print all key/value pairs to the file `list.txt` in the temporary directory. We redirect the user to another file.

```
class EX_CGI4

inherit

    EPX_CGI

    EPX_FACTORY

creation

    make

feature

    execute is
        do
            assert_key_value_pairs_created
            save_values

            extend ("Location: /mydir/myfile.html")
            new_line
            new_line
        end

    save_values is
        local
            fout: STDC_TEXT_FILE
            i: INTEGER
            kv: EPX_KEYVALUE
        do
            create fout.create_write (fs.temporary_directory + "/list.txt")
            from
                i := cgi_data.lower
            until
                i > cgi_data.upper
            loop
                kv := cgi_data.item (i)
                fout.puts (kv.key)
                fout.puts ("%T")
                fout.puts (kv.value)
                fout.puts ("%N")
                i := i + 1
```

```
            end
            fout.close
        end

    end
```

In this chapter:

- ***Compiling POSIX programs in Windows***
- ***Native Windows***
- ***Binary mode versus text mode***

8 e-POSIX in Windows

e-POSIX offers three alternatives to writing programs that run on both Unix and Windows platforms:

1. Write programs that only rely on Standard C. If you use only Standard C classes your program is probably quite portable. Standard C doesn't offer that much however.
2. Write programs that are based on POSIX. You use a POSIX emulator to compile and run your program unchanged on Windows. The only thing you have to be aware of is the distinction between binary and text files.
3. Write programs that are based upon e-POSIX's EPX_XXXX layer. This layer is based on e-POSIX's ABSTRACT_XXXX classes, that covers code that is common between Windows and a POSIX platform.
   Previous versions of e-POSIX used a factory class approach to access this common code. This is no longer needed. The ABSTRACT_XXXX are maded effective through EPX_XXXX classes when compiling for Windows or for POSIX.

The following sections offer more details about the last two approaches.

8.1 Compiling POSIX programs in Windows

You can also use a very large subset of POSIX under Windows with a POSIX emulator. I've tested this using SmallEiffel and Cygwin's freely available emulator. Here the steps:

1. Download the Cygwin toolkit from `http://sources.redhat.com/cygwin`.
2. Set the compiler in `compiler.se` to `gcc`. Leave the system in `system.se` to Windows.
3. Change the SEDIR variable in the `Makefile`. Make sure to use a Unix style path, for example `//c/elj-win32/SmallEiffel`.
4. Compile the library `libeposix.a` with:
   ```
   make libeposix.a
   ```
   This make should of course be the GNU make from Cygnus.
5. Now you can run a test, for example:
   ```
   make test
   ```
   This will select the last uncommented root classes and test it.

A few things are not available under Cygnus' POSIX emulation:

1. *POSIX_FILE_SYSTEM.create_fifo* is not supported. Any attempt to use it will return ENOSYS. I'm not sure if returning an error is the correct solution for applications that require POSIX compatibility, because you are only warned at run–time. Another solution would be to include a call to `mkfifo`and if you use it, let the linker complain.
2. There is no locking, so calls to *POSIX_FILE_DESCRIPTOR.get_lock* and such will fail.

3. Certain POSIX tests assume that a more Unix like environment is available, so not all tests will run. For example the standard Cygwin distribution doesn't have a `more` utility. If you make a symbolic link from `less` to `more` the child process test will run.
4. The current list of implemented functions is available from `http://sources.redhat.com/cygwin/faq/faq_3.html#SEC17`.

8.2 Native Windows

Previous versions of e-POSIX used a factory class approach to access Windows or POSIX specific code. This is obsolete.

If you want to write code that is portable between Windows and POSIX use the EPX_XXXX class layer. For example you can use the *EPX_FILE_DESCRIPTOR* to use file descriptors that are completely portable between these two OSes. Use *EPX_FILE_SYSTEM* to have access to file system specific code to change directories or get the temporary directory.

In general you can replace the POSIX_ prefix with EPX_ to compile most of the examples presented in the previous POSIX specific chapters. The classes currently available in the EPX_XXXX layer are:

- *EPX_CURRENT_PROCESS*.
- *EPX_EXEC_PROCESS*.
- *EPX_FILE_DESCRIPTOR*.
- *EPX_FILE_SYSTEM*.
- *EPX_PIPE*.

Figure one shows hoe the *EPX_FILE_DESCRIPTOR* class is derived from *ABSTRACT_FILE_DESCRIPTOR*. Both Windows and POSIX have an effective *EPX_FILE_DESCRIPTOR* class. Classes as *POSIX_FILE_DESCRIPTOR* implement POSIX specific functionality for a file descriptor.

An example of using the *EPX_FILE_SYSTEM* class is shown below:

```
class EX_EPX1

inherit

    EPX_FILE_SYSTEM

creation

    make

feature

    make is
        local
            dir: STRING
        do
            print ("Current directory: ")
            dir := current_directory
```

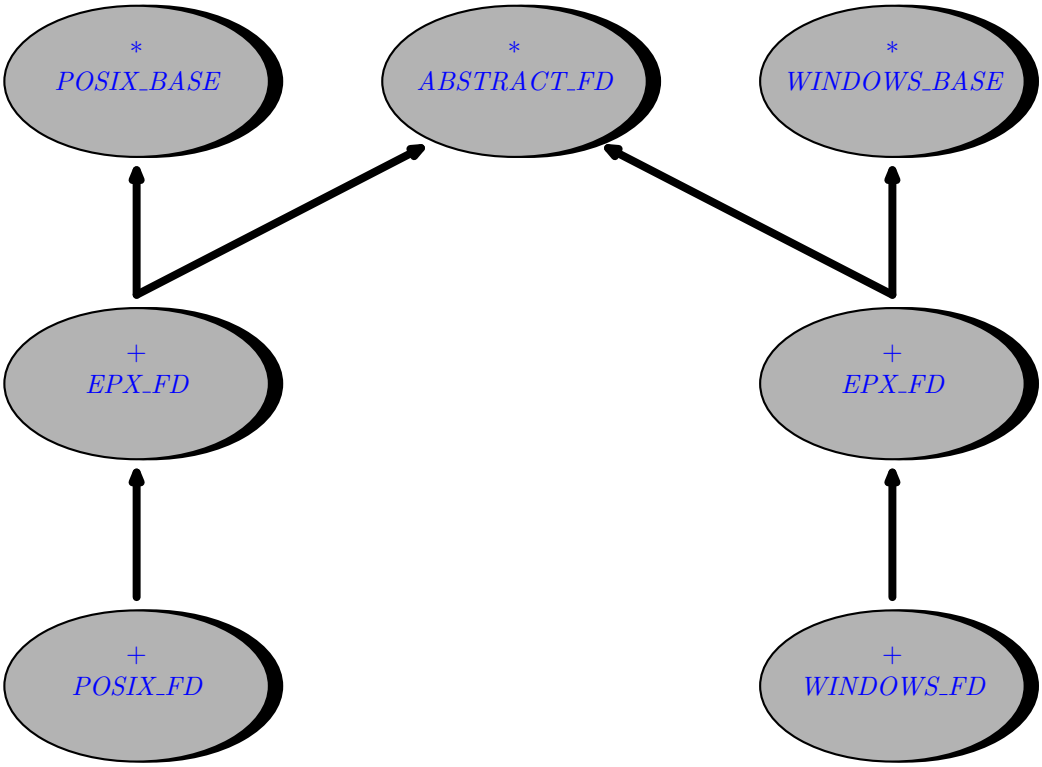

Figure 8.1 How EPX_XXXX classes are related to the POSIX and Windows classes

```
        print (dir)
        print ("%N")
        change_directory ("..")
        change_directory (dir)
        make_directory ("abc")
        rename_to ("abc", "def")
        remove_directory ("def")
    end

end
```

In **appendix C** all abstract classes are listed. There deferred features are made effective in the EPX class for the operating system you're compiling for.

8.3 Binary mode versus text mode

Independent of what layer you use to write Windows programs, you have to deal with binary and text modes. And if you usually write Unix programs and want them to work on Windows too, you have to bother with it too.

On Windows, each line of a text files ends with a carriage return character followed by a line feed character. In Eiffel this is the string `"%R%N"`. If you use streams, *STDC_TEXT_FILE* and *STDC_BINARY_FILE*, this distinction is transparently handled.

For file descriptions, it depends. If you use Cygwin, file descriptors can either be binary or text. This depends on Cygwin settings. If you use the EPX layer, file descriptors are binary, period. The reason for this is that the underlying code is the Win32 file API which is binary only. This is usually no problem. Reading a line will still stop when a `'%N'` is encountered. If you use the *ABSTRACT_FILE_DESCRIPTOR.chop_last_string* method, it automatically removes any `'%R'` if there was such a character just before the `'%N'`.

At this moment, there is no explicit support for creating text files using file descriptors with the proper Windows end of file characters. *STDC_TEXT_FILE* works fine here work fine here.

If you want to use binary standard input or binary standard output, use the file descriptors available in *EPX_CURRENT_PROCESS* as *fd_stdin* and *fd_stdout*.

For Cygwin users, the following information can be helpful to get the binary versus text file distinction correct:

- Mount the volume in binary mode.
- Set the environment variable CYGWIN to 'binary'.

More information about Cygwin and CR/LF handling can be found at `http://sources.redhat.com/cygwin/faq/faq_toc.html#TOC62`.

In this chapter:

- ***Error handling with exceptions***
- ***Manual error handling***

9 Error handling

This chapter describes certain error handling strategies that are possible with e-POSIX. Basically there are two strategies: using the Eiffel exception mechanism or doing the error handling all yourself.

9.1 Error handling with exceptions

The opinion of the author of e-POSIX is that Eiffel's exception mechanism is very well suited to deal with things like files that cannot be opened or directories that do not exist. Others disagree, see **section 9.2**. e-POSIX is designed such that when a POSIX routine returns an error code, an exception is thrown. Here my arguments why I favor this style of error handling:

1. We all know that exceptions are to be used for breach of contract. This idea is formulated in [5]and is the best expressed opinion of exception handling I know.
   So if you ask an e-POSIX method to open a file, it will do that for you. If it cannot open the file, for whatever reason, it will raise an exception. The same argument hold if you ask it to go to a directory, to start a program, or to open a connection to another machine.
   This approach is also reflected in the names of e-POSIX's features. The name is *POSIX_TEXT_FILE.open_read* and not *POSIX_TEXT_FILE.attempt_open_read*.
2. It is usually not wise to trust clients with error handling. The larger a distance between a software failure and the error report, the more difficult it is to make a correct diagnosis of what went wrong (see [6]). e-POSIX uses the fail early, fail hard approach.
3. Error handling is often forgotten or left to some global general error handling mechanism. In an interesting article (see [7]) James Whittaker describes how he modified certain system calls to return legitimate, but unexpected return codes. Memory allocation failed for example, or opening a file returned with no more file handles. Applications failed within seconds, but it was usually completely unclear why.
4. It's a lot easier for programmer's. You don't have to write any error handling. If your program completed, you know that there wasn't a single system call that failed, that you didn't continue despite some error. This will make it possible to write programs that do their work correctly if no errors occur, or else do nothing.

First an example. Let's take a look at the code you have to write in case you want to handle failure of opening a file:

```
[file ex_error1 does not exist]
```

In this example we try to create a file exclusively. The create will fail if the file already exists. In case this happens, we retry 3 times. Before retrying we wait 1 second. Note that if the error is not `EEXIST`, we fail directly, without retrying.

In my opinion above's code is just the code you want to write usually: do not worry about errors, if something goes wrong, your application will fail.

My preferred way of error handling is (or sometimes should be) also reflected in the preconditions. For example the *POSIX_FILE_SYSTEM.browse_directory* has the precondition that the given path should exist and should be a directory. Quite reasonable I think. The argument against such preconditions is that it is somewhat strange: if a client has honoured the precondition by checking that the directory exists, it should be able to assume that it safely can call the routine. But between its own check and the actual call, the directory can be removed by another process.

This is the concurrent precondition paradox (see [5]). In my opinion it would not be wise to remove this precondition. It is true that honouring it, will not make sure the contract is not broken. But it still serves a very usefull purpose: documentation.

For example the routine *POSIX_FILE_SYSTEM.remove_file* does not have the precondition that the file should exist. That isn't an oversight. This routine does not fail if the file no longer exists for good reason: it honours its postcondition after all. So when you call this routine, the file may or may not exist. The routine doesn't care.

9.2 Manual error handling

In spite of the arguments listed in the previous section, automatic error handling is perhaps tedious to use when you expect a lot of errors. And some programmers just do not like Eiffel's exception mechanism. Therefore e-POSIX implements a completely different style of error handling. In this case, e-POSIX continues when an error occurs, but it safes the errorcode, and you can check the errorcode of the first error when you wish. This first errorcode has to be reset by the programmer. An example:

```
[file ex_error2 does not exist]
```

Exception handling is turned off by a call to *STDC_SECURITY_ACCESSOR.security.error_handling.disable_exceptions*. It can be enabled again by calling *security.error_handling.enable_exceptions*. In between, you're on your own, just like a C programmer. If `myfile` cannot be opened, nothing happens, and the *POSIX_FILE_DESCRIPTOR.write_string* feature is called. Depending if you have enabled precondition checking or not, *write_string* will fail. The precondition if *write_string* is that the file has to be open. Therefore, at certain points, you're still forced to deal with errors. Every object has an *errno* variable. This variable points to the global *STDC_ERRNO* object (its a once routine). So there basically is just one *first_value* error value. Whatever object caused the error, you can check the *errno.first_value* of any e-POSIX object. The last error is still available in *errno.value*.

If there is no error, the program continues writing. If *POSIX_FILE_DESCRIPTOR.write_string* failed, the next one is still executed. If there is an error, we reset it with *STDC_ERRNO.clear_first*. This gives us the chance to catch another error value if an error occurs. If this method is not called, *first_value* will keep its original value.

The following example is the same as *EX_ERROR1*. It shows how to open a file exclusively with manual error handling.

```
[file ex_error3 does not exist]
```

As you can see, manual error handling does not necessarily translate into less code.

The summary of this section is that you chould check each distinctive step when using manual error handling. You don't have to check intermediate steps.

In this chapter:

- ***Denial of service attacks***
- ***Authorization bypass attacks***

10 Security

e-POSIX is well–suited to write server applications like CGI scripts and daemons. As these applications can be hosted on servers that are attached to the Internet, they could be prone to attack. Applications written with e-POSIX could be misused in a denial of service attack or to gain root access. e-POSIX offers certain protection mechanisms that enable your applications to fend off such penetrations.

This chapter shows you how applications can be misused and what mechanisms e-POSIX offers for certain attacks.

> “Programmers typically focus on "positive" aspects of programs, that is, what is the functionality required for the task to be accomplished. Programmers rarely focus on the negative aspects of programs, that is, what functionality is not required for the program to accomplish its task. Attackers take advantage of proggrammers failure to consider negative functionality. Perhaps a reason that programmers avoid negative functionality is that there is no good way to specify what a program should not be permitted to do.”

10.1 Denial of service attacks

In a denial of service attack, crackers attempt to deplete one or more finite resources. Resources can be software related like database connections or TCP/IP connections, but ultimately resources are finite because of hardware limitations. This manual distinguishes the following hardware resources:

- Memory.
- CPU.
- Disk space.
- Network bandwidth.

A denial of service attack succeeds if a cracker depletes these resources in such a way that the server cannot handle request anymore, or handles them very slowly. For example, Linux 2.2 is easy to bring to its knees if you keep on allocating memory. In normal situations your application runs fine, and allocates only a limited amount of memory. But an attacker might have found a way to make your application allocate much more memory. Even if you are sure that the code you have written is not prone to such an attack, you might use a library based on e-POSIX that does have code that is exploitable.

e-POSIX has some limited support to set limits on memory, file handle (a memory issue) and cpu usage. When a set limit has been exceeded, an exception is raised.

To limit the amount of memory, inherit from *STDC_SECURITY_ACCESSOR* and call *security.memory.set_max_allocation*. Currently this limits the amount of memory that can be allocated with *STDC_BUFFER*. It does not limit the amount of memory that is allocated by *STRING* or

other classes. You can also limit the amount of memory that can be allocated with a single call by calling *security.memory.set_max_single_allocation*.

You can limit the number of file handles a program can open by calling *security.files.set_max_open_files*. This works only with files and sockets opened by e-POSIX classes as *STDC_FILE* and *POSIX_FILE_DESCRIPTOR*, not with files opened through other means. In this case you cannot rely on the garbage collection to close your file. Certain garbage collectors do not allow calling other classes in the *MEMORY.dispose* method. e-POSIX needs to do this to decrement its idea of the number of open handles. Only when you explicitly call *STDC_FILE.close* will the e-POSIX decrease its open file handles.

You can limit the amount of CPU time by calling *security.cpu.set_max_process_time*. It is not possible to automatically halt your application when this time has exceeded. You have to call *security.cpu.check_process_time* to actually check the processor time used.

Currently e-POSIX cannot check disk space or network bandwidth limitations.

10.2 Authorization bypass attacks

A hacker can bypass authorization if he or she, through your program, can gain the following access:

- Access to more information than your program is written to provide. Security is not breached here, but your program is used in an ‘innovative’ way. Note that if your program runs within the root security context (suid root), security can be breached!
- Security is breached when your program is used to get more access rights than your program is written to provide. Especially suid root programs are an attractive target here.

Usually Eiffel programs do not allocate buffers on the stack, so they are not prone to the so called ‘buffer overflow’ attack. As certain vendors might provide some ‘native’ class that allocate things on the stack, leave precondition checking always on in suid root programs.

Currently e-POSIX doesn’t offer much protection for suid root programs. Much better security will be the topic of a next release.

In this chapter:

- ***Making C Headers available to Eiffel***
- ***Distinction between Standard C and POSIX headers***
- ***C translation details***

11 Accessing C headers

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

11.1 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`.

11.2 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 *CAPI_STDIO*.

In **table 11.1** all the groups with there translation header prefix and Eiffel class prefix are listed. See also the directory structure in **figure 11.1**.

11.3 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.

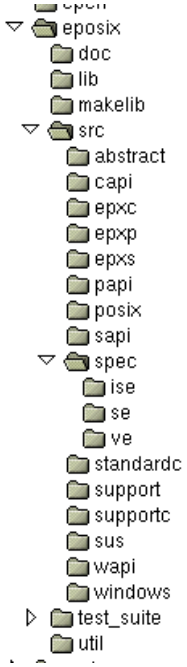

Figure 11.1 e-POSIX directory structure

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 11.1 e-POSIX prefix conventions

In this chapter:

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>`	*ABSTRACT_FILE_SYSTEM.is_accessible*	
`aio_cancel`	`<aio.h>`	*POSIX_ASYNC_IO_REQUEST.cancel*	
`aio_error`	`<aio.h>`	*POSIX_ASYNC_IO_REQUEST.is_pending*	
`aio_fsync`	`<aio.h>`	*POSIX_ASYNC_IO_REQUEST.synchronize*	
`aio_read`	`<aio.h>`	*POSIX_ASYNC_IO_REQUEST.read*	
`aio_return`	`<aio.h>`	*POSIX_ASYNC_IO_REQUEST.return_status*	
`aio_suspend`	`<aio.h>`	*POSIX_ASYNC_IO_REQUEST.wait_for*	
`aio_write`	`<aio.h>`	*POSIX_ASYNC_IO_REQUEST.write*	
`alarm`	`<unistd.h>`	*POSIX_TIMED_COMMAND*	
`asctime`	`<time.h>`	*STDC_TIME.default_format*	
`atexit`	`<stdlib.h>`		probably not applicable.
`calloc`	`<stdlib.h>`	*STDC_BUFFER.allocate_and_clear*	
`cfgetispeed`	`<termios.h>`	*POSIX_TERMIOS.input_speed*	
`cfgetospeed`	`<termios.h>`	*POSIX_TERMIOS.output_speed*	
`cfsetispeed`	`<termios.h>`	*POSIX_TERMIOS.set_input_speed*	
`cfsetospeed`	`<termios.h>`	*POSIX_TERMIOS.set_output_speed*	
`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>`	*STDC_CURRENT_PROCESS.clock*	
`clock_getres`	`<time.h>`		
`clock_gettime`	`<time.h>`		
`clock_settime`	`<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>`		Can be emulated with *STDC_TIME*.
`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.
`fdatasync`	`<unistd.h>`	*POSIX_FILE_DESCRIPTOR.synchronize_data*	This function is not available on many so called POSIX systems. In such cases it is mapped to `fsync`.
`fdopen`	`<stdio.h>`	*POSIX_FILE.make_from_file_descriptor*	
`feof`	`<stdio.h>`	*STDC_FILE.eof*	
`ferror`	`<stdio.h>`	*STDC_FILE.error*	
`fflush`	`<stdio.h>`	*STDC_FILE.flush*	
`fgetc`	`<stdio.h>`	*STDC_FILE.get_character*	
`fgetpos`	`<stdio.h>`	*STDC_FILE.get_position*	
`fgets`	`<stdio.h>`	*STDC_FILE.get_string*	
`fileno`	`<stdio.h>`	*POSIX_FILE_DESCRIPTOR.make_from_file*	
`fopen`	`<stdio.h>`	*STDC_FILE*	various open creation features.
`fork`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.fork*	
`fpathconf`	`<unistd.h>`		
`fprintf`	`<stdio.h>`		not applicable.
`fputc`	`<stdio.h>`	*STDC_FILE.putc*	
`fputs`	`<stdio.h>`	*STDC_FILE.put_string*	
`fread`	`<stdio.h>`	*STDC_FILE.read*	Also *read_string* and *read_character*.
`free`	`<stdlib.h>`	*STDC_BUFFER.deallocate*	
`freopen`	`<stdio.h>`	*STDC_FILE.reopen*	
`fseek`	`<stdio.h>`	*STDC_FILE.seek*	Also *seek_from_current* and *seek_from_end*.
`fsetpos`	`<stdio.h>`	*STDC_FILE.set_position*	
`fstat`	`<sys/stat.h>`	*POSIX_STATUS*	Returned by *POSIX_FILE_DESCRIPTOR.status*.
`fsync`	`<unistd.h>`	*POSIX_FILE_DESCRIPTOR.synchronize*	
`ftell`	`<stdio.h>`	*STDC_FILE.tell*	
`fwrite`	`<stdio.h>`	*STDC_FILE.write*	
`getc`	`<stdioh>`		See `fgetc`.
`getchar`	`<stdio.h>`		See `fgetc`.
`getcwd`	`<unistd.h>`	*POSIX_FILE_SYSTEM.current_directory*	
`getegid`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.effective_group_id*	
`getenv`	`<stdlib.h>`	*STDC_ENV_VAR.value*	
`geteuid`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.effective_user_id*	
`getgid`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.real_group_id*	
`getgrgid`	`<grp.h>`	*POSIX_GROUP.make_from_gid*	
`getgrnam`	`<grp.h>`	*POSIX_GROUP.make_from_name*	
`getgroups`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.is_in_group*	
`getlogin`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.login_name*	
`getpgrp`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.process_group_id*	
`getpid`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.pid*	
`getppid`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.parent_pid*	
`getpwnam`	`<pwd.h>`	*POSIX_USER.make_from_name*	
`getpwuid`	`<pwd.h>`	*POSIX_USER.make_from_uid*	
`gets`	`<stdio.h>`		See `fgets`.
`getuid`	`<unistd.h>`	*POSIX_CURRENT_PROCESS.real_user_id*	
`gmtime`	`<time.h>`	*STDC_TIME.to_utc*	
`isatty`	`<unistd.h>`	*POSIX_FILE_DESCRIPTOR.is_attached_to_terminal*	
`kill`	`<signal.h>`	*POSIX_PROCESS.kill*	
`link`	`<unistd.h>`	*POSIX_FILE_SYSTEM.link*	
`lio_listio`	`<aio.h>`		

localeconv	<locale.h>	*STDC_LOCALE_NUMERIC*	
localtime	<time.h>	*STDC_TIME.to_local*	
lseek	<unistd.h>	*POSIX_FILE_DESCRIPTOR.seek*	Also *seek_from_current* and *seek_from_end*.
malloc	<stdlib.h>	*STDC_BUFFER.allocate*	
memcpy	<string.h>	*STDC_BUFFER.memory_copy*	See also *copy_from*.
memchr	<string.h>		
memcmp	<string.h>		
memmove	<string.h>	*STDC_BUFFER.memory_move*	
memset	<string.h>	*STDC_BUFFER.fill_with*	
mkdir	<sys/stat.h>	*POSIX_FILE_SYSTEM.make_directory*	
mkfifo	<sys/stat.h>	*POSIX_FILE_SYSTEM.create_fifo*	
mktime	<time.h>	*STDC_TIME.set_date_time*	Also *set_date* and *set_time*.
mlockall	<sys/mman.h>		
mlock	<sys/mman.h>		
mmap	<sys/mman.h>	*POSIX_MEMORY_MAP*	
mprotect	<sys/mman.h>		
mq-receive	<mqueue.h>		
mq_close	<mqueue.h>		
mq_getattr	<mqueue.h>		
mq_notify	<mqueue.h>		
mq_open	<mqueue.h>		
mq_send	<mqueue.h>		
mq_setattr	<mqueue.h>		
mq_unlink	<mqueue.h>		
msync	<sys/mman.h>		
munlockall	<sys/mman.h>		
munlock	<sys/mman.h>		
munmap	<sys/mman.h>	*POSIX_MEMORY_MAP*	
nanosleep	<time.h>		
open	<fcntl.h>	*POSIX_FILE_DESCRIPTOR.open*	Also *open_read*, *open_read_write* and *open_write*
opendir	<dirent.h>	*POSIX_DIRECTORY*	
pathconf	<unistd.h>	*POSIX_DIRECTORY.max_filename_length*	
pause	<unistd.h>	*POSIX_CURRENT_PROCESS.pause*	
perror	<stdio.h>		e-POSIX generates exceptions on error.
pipe	<unistd.h>	*POSIX_PIPE.make*	
printf	<stdio.h>		not applicable.
putc	<stdio.h>		See `fputc`.
putchar	<stdio.h>		See `fputc`.
puts	<stdio.h>		See `fputs`.
raise	<signal.h>	*STDC_SIGNAL.raise*	
rand	<stdlib.h>	*STDC_CURRENT_PROCESS.random*	
read	<unistd.h>	*POSIX_FILE_DESCRIPTOR.read*	
readdir	<dirent.h>	*POSIX_DIRECTORY*	
realloc	<stdlib.h>	*STDC_BUFFER.resize*	
remove	<stdio.h>	*POSIX_FILE_SYSTEM.remove_file*	
rename	<unistd.h>	*POSIX_FILE_SYSTEM.rename_to*	
rewind	<stdio.h>	*STDC_FILE.rewind*	
rewinddir	<dirent.h>	*POSIX_DIRECTORY*	
rmdir	<unistd.h>	*POSIX_FILE_SYSTEM.remove_directory*	
scanf	<stdio.h>		not applicable.
sem_close	<semaphore.h>		
sem_destroy	<semaphore.h>		
sem_getvalue	<semaphore.h>		
sem_init	<semaphore.h>	*POSIX_UNNAMED_SEMAPHORE.create_shared*	And *create_unshared*.
sem_open	<semaphore.h>		
sem_post	<semaphore.h>	*POSIX_SEMAPHORE.release*	

sem_trywait	<semaphore.h>	*POSIX_SEMAPHORE.attempt_acquire*	
sem_unlink	<semaphore.h>		
sem_wait	<semaphore.h>	*POSIX_SEMAPHORE.acquire*	
setbuf	<stdio.h>	*STDC_FILE.set_buffer*	
setgid	<unistd.h>	*POSIX_CURRENT_PROCESS.set_group_id*	Also *restore_group_id*.
setlocale	<locale.h>	*STDC_CURRENT_PROCESS.set_locale*	Also *set_native_locale* and *set_native_time*.
setpgid	<unistd.h>	*PAPI_UNISTD.posix_setsid*	
setsid	<unistd.h>	*PAPI_UNISTD.posix_setsid*	
setuid	<unistd.h>	*POSIX_CURRENT_PROCESS.set_user_id*	Also *restore_user_id*.
setvbuf	<stdio.h>	*STDC_FILE.set_no_buffering*	Also *set_full_buffering* and *set_line_buffering*
shm_open	<sys/mman.h>	*POSIX_SHARED_MEMORY.open_read_write*	And *create_write*, *open_read*, Efeatureopen_write.
shm_unlink	<sys/mman.h>	*POSIX_FILE_SYSTEM.unlink_shared_memory_object*	
sigaction	<signal.h>	*POSIX_SIGNAL*	
sigaddset	<signal.h>	*POSIX_SIGNAL_SET.add*	
sigdelset	<signal.h>	*POSIX_SIGNAL_SET.prune*	
sigemptyset	<signal.h>	*POSIX_SIGNAL_SET.make_empty*	
sigfillset	<signal.h>	*POSIX_SIGNAL_SET.make_full*	
sigismember	<signal.h>	*POSIX_SIGNAL_SET.has*	
signal	<signal.h>	*STDC_SIGNAL.raise*	
sigpending	<signal.h>	*POSIX_SIGNAL_SET.make_pending*	
sigprocmask	<signal.h>	*POSIX_SIGNAL_SET.add_to_blocked_signals*	Also *remove_from_blocked_signals* and *set_blocked_signals*
sigqueue	<signal.h>		
sigsuspend	<signal.h>	*POSIX_SIGNAL_SET.suspend*	
sigtimedwait	<signal.h>		
sigwait	<signal.h>		
sigwaitinfo	<signal.h>		
sleep	<unistd.h>	*POSIX_CURRENT_PROCESS.sleep*	
sprintf	<stdio.h>		Not applicable.
srand	<stdlib.h>	*STDC_CURRENT_PROCESS.set_random_seed*	
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>	*POSIX_TERMIOS.flush_input*	
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*	
timer_create	<signal.h>		
timer_create	<time.h>		
times	<times.h>		
tmpfile	<stdio.h>	*STDC_TEMPORARY_FILE.make*	
tmpnam	<stdio.h>	*STDC_FILE_SYSTEM.temporary_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.
vsprint	<stdio.h>		Not applicable.
wait	<sys/wait.h>	*POSIX_CURRENT_PROCESS.wait*	
waitpid	<sys/wait.h>	*POSIX_FORK_ROOT.wait_pid*	
write	<unistd.h>	*POSIX_FILE_DESCRIPTOR.write*	

This tabel does not contain the following category of functions:

1. Math functions.
2. String functions, including wide character/multibyte string routines. The memory move/copy functions are included, some of them even supported.
3. No type conversion functions.
4. No functions from `<ctype.h>`.
5. No functions from `<setjmp.h>`.
6. No functions from `<stdarg.h>`.
7. No string formatting functions like `sscanf`. I suggest you use the Formatter library for that. You can download this excellent library at `http://www.eiffel-forum.org/archive/dominicu/format.htm`.

Functions in above categories are either not applicable, already present in Eiffel or are better off in a different library.

B
Short (flat) listing of Standard C classes

In this chapter:

B.1 STDC_BASE

```
class interface STDC_BASE
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
invariant
    accessing_real_singleton: security_is_real_singleton;
end of STDC_BASE
```

B.2 STDC_BUFFER

```
class interface STDC_BUFFER
creation
    allocate (a_capacity: INTEGER)
        -- allocate memory of a_capacity bytes
        require
            valid_capacity: a_capacity > 0;
            allowed: security.memory.is_allowed_allocation(a_capacity)
        ensure
            successfull_allocation: is_allocated
    allocate_and_clear (a_capacity: INTEGER)
        -- allocate memory of a_capacity bytes, make sure its zeroed out
        require
            valid_capacity: a_capacity > 0;
            allowed: security.memory.is_allowed_allocation(a_capacity)
        ensure
            successfull_allocation: is_allocated
    make_from_pointer (a_pointer: POINTER; a_capacity: INTEGER; become_owner: BOOLEAN)
        -- attach a pointer to this object. If become_owner is
        -- True, it will deallocate the pointer!
        require
            valid_capacity: a_capacity >= 0;
            valid_pointer: a_pointer /= default_pointer
feature(s) from STDC_BUFFER
    -- creation
    allocate (a_capacity: INTEGER)
        -- allocate memory of a_capacity bytes
        require
            valid_capacity: a_capacity > 0;
            allowed: security.memory.is_allowed_allocation(a_capacity)
        ensure
            successfull_allocation: is_allocated
    allocate_and_clear (a_capacity: INTEGER)
        -- allocate memory of a_capacity bytes, make sure its zeroed out
        require
            valid_capacity: a_capacity > 0;
            allowed: security.memory.is_allowed_allocation(a_capacity)
        ensure
            successfull_allocation: is_allocated
    make_from_pointer (a_pointer: POINTER; a_capacity: INTEGER; become_owner: BOOLEAN)
        -- attach a pointer to this object. If become_owner is
        -- True, it will deallocate the pointer!
        require
            valid_capacity: a_capacity >= 0;
            valid_pointer: a_pointer /= default_pointer
feature(s) from STDC_BUFFER
```

```
    -- other allocation commands
    deallocate
        -- free the allocated memory now, dont wait for garbage collector.
        require
            may_deallocate: is_owner;
            not_deallocated: is_allocated
        ensure
            now_deallocated: not is_allocated
    detach
        -- this object is no longer the owner of the allocated memory block
        require
            not_deallocated: is_allocated
        ensure
            looks_deallocated: not is_allocated
    resize (new_capacity: INTEGER)
        -- resize memory to new_capacity bytes. Expanded memory is not
        -- guaranteed to be zeroed out.
        require
            valid_capacity: new_capacity > 0;
            allocated: is_allocated;
            may_resize: is_owner
        ensure
            successfull_allocation: is_allocated;
            enough_capacity: capacity >= new_capacity
    realloc (new_capacity: INTEGER)
        -- resize memory to new_capacity bytes. Expanded memory is not
        -- guaranteed to be zeroed out.
        require
            valid_capacity: new_capacity > 0;
            allocated: is_allocated;
            may_resize: is_owner
        ensure
            successfull_allocation: is_allocated;
            enough_capacity: capacity >= new_capacity
feature(s) from STDC_BUFFER
    -- copy data internally or externally
    copy_from (source: STDC_BUFFER; src_offset, dest_offset, bytes: INTEGER)
        -- Move data from another buffer into ourselves.
        -- Start at offset src_offset, into
        -- offset dest_offset, moving bytes bytes
        -- memory may overlap
        require
            valid_source: source /= Void;
            valid_source_offset: src_offset >= 0 and src_offset + bytes <= source.capacity;
            valid_dest_offset: dest_offset >= 0 and dest_offset + bytes <= capacity;
            valid_bytes_to_move: bytes >= 0
    memory_copy (source: POINTER; src_offset: INTEGER; dest_offset, bytes: INTEGER)
```

```
        -- Copy data from source, offset src_offset, to location
        -- dest_offset in this buffer, for bytes bytes
        -- memory may not overlap, use move to copy within buffer
        -- or memory_move to copy from potentially overlapping buffer
        require
            my_memory_does_not_overlap: true;
            I_know_what_I_am_doing: true;
            valid_source: source /= default_pointer;
            data_should_fit: dest_offset + bytes <= capacity;
            valid_bytes_to_move: bytes >= 0
    memcpy (source: POINTER; src_offset: INTEGER; dest_offset, bytes: INTEGER)
        -- Copy data from source, offset src_offset, to location
        -- dest_offset in this buffer, for bytes bytes
        -- memory may not overlap, use move to copy within buffer
        -- or memory_move to copy from potentially overlapping buffer
        require
            my_memory_does_not_overlap: true;
            I_know_what_I_am_doing: true;
            valid_source: source /= default_pointer;
            data_should_fit: dest_offset + bytes <= capacity;
            valid_bytes_to_move: bytes >= 0
    memory_move (source: POINTER; src_offset: INTEGER; dest_offset, bytes: INTEGER)
        -- Copy data from source, offset src_offset, to location
        -- dest_offset in this buffer, for bytes bytes
        require
            valid_source: source /= default_pointer;
            data_should_fit: dest_offset + bytes <= capacity;
            valid_bytes_to_move: bytes >= 0
    memmove (source: POINTER; src_offset: INTEGER; dest_offset, bytes: INTEGER)
        -- Copy data from source, offset src_offset, to location
        -- dest_offset in this buffer, for bytes bytes
        require
            valid_source: source /= default_pointer;
            data_should_fit: dest_offset + bytes <= capacity;
            valid_bytes_to_move: bytes >= 0
    move (src_offset, dest_offset: INTEGER; bytes: INTEGER)
        -- Move data around in buffer itself from offset src_offset to
        -- offset dest_offset, moving bytes bytes
        -- memory may overlap
        require
            valid_source_offset: src_offset >= 0 and src_offset + bytes <= capacity;
            valid_dest_offset: dest_offset >= 0 and dest_offset + bytes <= capacity
feature(s) from STDC_BUFFER
    -- set/get bytes (8-bit data)
    peek_uint8 (index: INTEGER): INTEGER
        -- consider memory an array of 8 bit values.
        require
```

```
            valid_index: is_valid_index(index)
        ensure
            possible_values: Result >= 0 and Result < 256
feature(s) from STDC_BUFFER
    -- set/get bytes (8-bit data)
    infix "@" (index: INTEGER): INTEGER
        -- consider memory an array of 8 bit values.
        require
            valid_index: is_valid_index(index)
        ensure
            possible_values: Result >= 0 and Result < 256
    poke_uint8 (index, value: INTEGER)
        require
            valid_index: is_valid_index(index);
            valid_byte: value >= 0 and value < 256
        ensure
            consistent: peek_uint8(index) = value
    peek_int8 (index: INTEGER): INTEGER
        -- consider memory an array of 8 bit values.
        require
            valid_index: is_valid_index(index)
        ensure
            possible_values: Result >= - 128 and Result <= 127
    poke_int8 (index, value: INTEGER)
        require
            valid_index: is_valid_index(index);
            valid_tinyin: value >= - 128 and value <= 127
        ensure
            consistent: peek_int8(index) = value
feature(s) from STDC_BUFFER
    -- set/get integers (16-bit data)
    peek_int16 (index: INTEGER): INTEGER
        -- consider memory an array of signed 16 bit values.
        require
            valid_index: is_valid_range(index,index + 1)
        ensure
            valid_result: Result >= - 32768 and Result <= 32767
feature(s) from STDC_BUFFER
    -- set/get integers (16-bit data)
    peek_int16_native (index: INTEGER): INTEGER
        -- consider memory an array of signed 16 bit values.
        require
            valid_index: is_valid_range(index,index + 1)
        ensure
            valid_result: Result >= - 32768 and Result <= 32767
    peek_uint16 (index: INTEGER): INTEGER
        -- consider memory an array of unsigned 16 bit values.
```

```
        require
            valid_index: is_valid_range(index,index + 1)
        ensure
            valid_result: Result >= 0 and Result <= 65535
    peek_uint16_native (index: INTEGER): INTEGER
        -- consider memory an array of unsigned 16 bit values.
        require
            valid_index: is_valid_range(index,index + 1)
        ensure
            valid_result: Result >= 0 and Result <= 65535
    peek_int16_big_endian (index: INTEGER): INTEGER
        -- consider memory an array of 16 bit values.
        require
            valid_index: is_valid_range(index,index + 1)
        ensure
            valid_result: Result >= - 32768 and Result <= 32767
    peek_int16_little_endian (index: INTEGER): INTEGER
        -- consider memory an array of 16 bit values.
        require
            valid_index: is_valid_range(index,index + 1)
        ensure
            valid_result: Result >= - 32768 and Result <= 32767
    poke_int16 (index: INTEGER; value: INTEGER)
        -- Write 16 bit value at offset index, in native endian format.
        require
            valid_index: is_valid_range(index,index + 1);
            valid_value: value >= - 32768 and value <= 32767
        ensure
            consistent: peek_int16_native(index) = value
    poke_int16_native (index: INTEGER; value: INTEGER)
        -- Write 16 bit value at offset index, in native endian format.
        require
            valid_index: is_valid_range(index,index + 1);
            valid_value: value >= - 32768 and value <= 32767
        ensure
            consistent: peek_int16_native(index) = value
    poke_int16_big_endian (index: INTEGER; value: INTEGER)
        -- Write 16 bit value at offset index, in big endian format.
        require
            valid_index: is_valid_range(index,index + 1);
            valid_value: value >= - 32768 and value <= 32767
        ensure
            consistent: peek_int16_big_endian(index) = value
    poke_int16_little_endian (index: INTEGER; value: INTEGER)
        -- Write 16 bit value at offset index, in little endian format.
        require
            valid_index: is_valid_range(index,index + 1);
```

```
            valid_value: value >= - 32768 and value <= 32767
        ensure
            consistent: peek_int16_little_endian(index) = value
feature(s) from STDC_BUFFER
    -- set/get integers (32-bit data)
    peek_int32_native (index: INTEGER): INTEGER
        -- Read 32 bit value at offset index, assume its byte order
        -- is native, and return it.
        require
            valid_index: is_valid_range(index,index + 3)
feature(s) from STDC_BUFFER
    -- set/get integers (32-bit data)
    peek_integer (index: INTEGER): INTEGER
        -- Read 32 bit value at offset index, assume its byte order
        -- is native, and return it.
        require
            valid_index: is_valid_range(index,index + 3)
    peek_int32_big_endian (index: INTEGER): INTEGER
        -- Read 32 bit value at offset index, assume its byte order
        -- is big endian, and return it in native format.
        require
            valid_index: is_valid_range(index,index + 3)
    peek_int32_little_endian (index: INTEGER): INTEGER
        -- Read 32 bit value at offset index, assume its byte order
        -- is little endian, and return it in native format.
        require
            valid_index: is_valid_range(index,index + 3)
    peek_uint32_native (index: INTEGER): INTEGER
        -- Read 32 bit unsigned int at offset index, assume native
        -- byte order.
        require
            valid_index: is_valid_range(index,index + 3)
    peek_uint32_big_endian (index: INTEGER): INTEGER
        -- Read 32 bit unsigned int at offset index, assume its
        -- byte order is big endian, and return it in native format.
        require
            valid_index: is_valid_range(index,index + 3)
    peek_uint32_little_endian (index: INTEGER): INTEGER
        -- Read 32 bit unsigned int at offset index, assume its
        -- byte order is big endian, and return it in native format.
        require
            valid_index: is_valid_range(index,index + 3)
    poke_integer (index: INTEGER; value: INTEGER)
        -- Write 32 bit value at offset index, in native endian format.
        require
            valid_index: is_valid_range(index,index + 3)
        ensure
```

```
            consistent: peek_integer(index) = value
    poke_int32_native (index: INTEGER; value: INTEGER)
        -- Write 32 bit value at offset index, in native endian format.
        require
            valid_index: is_valid_range(index,index + 3)
        ensure
            consistent: peek_integer(index) = value
    poke_int32_big_endian (index: INTEGER; value: INTEGER)
        -- Write 32 bit value at offset index, in big endian format.
        require
            valid_index: is_valid_range(index,index + 3)
        ensure
            consistent: peek_int32_big_endian(index) = value
    poke_int32_little_endian (index: INTEGER; value: INTEGER)
        -- Write 32 bit value at offset index, in little endian format.
        require
            valid_index: is_valid_range(index,index + 3)
        ensure
            consistent: peek_int32_little_endian(index) = value
feature(s) from STDC_BUFFER
    -- set/get characters
    peek_character (index: INTEGER): CHARACTER
        -- return value at position index as a character
        require
            valid_index: is_valid_index(index)
    poke_character (index: INTEGER; value: CHARACTER)
        -- set character at position index to value
        require
            valid_index: is_valid_index(index)
        ensure
            consistent: peek_character(index) = value
    substring (start_index, end_index: INTEGER): STRING
        -- Create a substring containing all characters
        -- from start_index to end_index inclusive.
        require
            valid_index: is_valid_range(start_index,end_index)
        ensure
            substring_not_void: Result /= Void;
            substring_count_as_asked: Result.count = end_index - start_index + 1
feature(s) from STDC_BUFFER
    -- fill
    fill_at (start_index, a_count: INTEGER; byte: INTEGER)
        -- Starting at position start_index, write byte for a_count bytes
        require
            valid_range: is_valid_range(start_index,start_index + a_count - 1);
            valid_byte: byte >= 0 and byte <= 255
feature(s) from STDC_BUFFER
```

```
    -- searching
    locate_character (other: CHARACTER; start_index: INTEGER): INTEGER
        ensure
            valid_result: Result >= - 1 and Result < capacity
    locate_string (other: STRING; start_index: INTEGER): INTEGER
        -- Does buffer contain other?
        -- Returns index where other is found.
        -- Returns -1 if not found
        -- searching starts at position start_index
        require
            other_not_empty: other /= Void and then not other.is_empty;
            valid_index: is_valid_index(start_index)
        ensure
            false_if_too_small: capacity < other.count implies Result = - 1;
            valid_result: Result >= - 1 and Result < capacity
feature(s) from STDC_BUFFER
    -- queries
    is_allocated: BOOLEAN
    is_valid_index (index: INTEGER): BOOLEAN
    is_valid_range (from_index, to_index: INTEGER): BOOLEAN
        -- Returns True if from_index..to_index is a valid and
        -- meaningfull range
feature(s) from STDC_BUFFER
    -- state
    capacity: INTEGER
        -- in number of bytes
    is_owner: BOOLEAN
        -- True if this object may deallocate the memory
    ptr: POINTER
        -- the actual pointer
feature(s) from STDC_BUFFER
    -- cleanup
    dispose
        -- Action to be executed just before garbage collection reclaims an
        -- object. (The default action is to do nothing at all.) If you want
        -- to change the default action, your class is supposed to
        -- inherit MEMORY and to redefine this dispose feature.
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_capacity: capacity >= 0;
    capacity_and_ptr_relation: (capacity = 0 implies not is_allocated) and capacity > 0 implies is_allocated;
end of STDC_BUFFER
```

B.3 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*
-- standard streams
stream_stdin: *POINTER*
stream_stdout: *POINTER*
stream_stderr: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
-- characters
const_EOF: *INTEGER*
-- signals EOF
feature(*s*) **from** *STDC_CONSTANTS*
-- buffering
IOFBF: *INTEGER*
-- full buffering
IOLBF: *INTEGER*
-- line buffering
IONBF: *INTEGER*
-- no buffering
feature(*s*) **from** *STDC_CONSTANTS*
-- file positioning
SEEK_SET: *INTEGER*
SEEK_CUR: *INTEGER*
SEEK_END: *INTEGER*
feature(*s*) **from** *STDC_CONSTANTS*
-- Signal related constants
SIG_DFL: *POINTER*
SIG_ERR: *POINTER*
SIG_IGN: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
-- Signals
SIGABRT: *INTEGER*
SIGFPE: *INTEGER*
-- erroneous arithmetic operation, such as zero divide or an
-- operation resulting in overflow
SIGILL: *INTEGER*
-- illegal instruction
SIGINT: *INTEGER*
-- receipt of an interactive attention signal
SIGSEGV: *INTEGER*

```
        -- invalid access to storage
    SIGTERM: INTEGER
feature(s) from STDC_CONSTANTS
    -- random numbers
    RAND_MAX: INTEGER
        -- maximum value returned by the random function
feature(s) from STDC_CONSTANTS
    -- category constants
    LC_CTYPE: INTEGER
    LC_NUMERIC: INTEGER
    LC_TIME: INTEGER
    LC_COLLATE: INTEGER
    LC_MONETARY: INTEGER
    LC_ALL: INTEGER
feature(s) from STDC_CONSTANTS
    -- various
    CLOCKS_PER_SEC: INTEGER
feature(s) from STDC_CONSTANTS
    -- exit codes
    EXIT_FAILURE: INTEGER
        -- exit status when something has gone wrong
    EXIT_SUCCESS: INTEGER
        -- exit status upon success
end of STDC_CONSTANTS
```

B.4 STDC_CURRENT_PROCESS

```
class interface STDC_CURRENT_PROCESS
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from STDC_CURRENT_PROCESS
    -- my stdandard input/output/error
    stdin: STDC_TEXT_FILE
    stdout: STDC_TEXT_FILE
    stderr: STDC_TEXT_FILE
invariant
    accessing_real_singleton: security_is_real_singleton;
end of STDC_CURRENT_PROCESS
```

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
    exist: BOOLEAN
        -- returns True if this environment variable is defined
    name: STRING
    value: STRING
        ensure
            -- contents of environment variable name if it exists
            -- or else the empty string
            not_void: Result /= Void
invariant
    accessing_real_singleton: security_is_real_singleton;
end of STDC_ENV_VAR
```

B.6 STDC_FILE

```
class interface STDC_TEXT_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
    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
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
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
        require
            open: is_open
        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
        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
        ensure
            valid_string: last_string.count <= bytes;
            read_consistent: last_read <= bytes
    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
        ensure
            valid_string: last_string.count <= bytes;
            read_consistent: last_read <= bytes
    read (buf: POINTER; offset, bytes: INTEGER)
        -- read chunk, set last_read
        require
            is_opened: is_open;
```

```
            valid_buf: buf /= default_pointer;
            valid_bytes: bytes >= 0
        ensure
            successfull_read: not eof implies last_read > 0;
            read_consistent: last_read <= bytes
feature(s) from STDC_FILE
    -- write, C like
    putc (c: INTEGER)
        -- write a single character
        ensure
            need_flush_set: need_flush
feature(s) from STDC_FILE
    -- write, C like
    put_character (c: INTEGER)
        -- write a single character
        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; offset, bytes: INTEGER)
        -- write bytes bytes from buf at offset offset
        -- we do not really care if offset is positive or negative...
        require
            is_opened: is_open;
            valid_buf: buf /= default_pointer;
            valid_bytes: bytes >= 0
        ensure
            successfull_write: last_written = bytes;
            need_flush_set: need_flush
feature(s) from STDC_FILE
    -- read, Eiffel like
    last_read: INTEGER
        -- last read bytes by some read_XXXX or get_string call
    last_boolean: BOOLEAN
        -- last boolean read by read_boolean
    last_character: CHARACTER
        -- last character read by read_character
    last_double: DOUBLE
        -- last double lread by read_double
    last_integer: INTEGER
    last_real: REAL
        -- last real read by read_real
    last_string: STRING
        -- last string read by read_string or get_string
    read_boolean
```

```
        -- attempt to read back a boolean written by write_boolean
    read_buffer (buf: STDC_BUFFER; offset, bytes: INTEGER)
        -- more safe version of read in case you have a
        -- STDC_BUFFER object.
        -- Check last_read for number of bytes actually read.
        require
            dont_overflow: buf.is_valid_index(offset + bytes - 1)
    read_double
    read_character
        -- read a single character and set last_character
        -- if end-of-file encountered, eof is True
    read_integer
    read_real
    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 if bytes
        -- are longer then the length of the actual line!
feature(s) from STDC_FILE
    -- write, Eiffel like
    last_written: INTEGER
        -- last written bytes by some write_XXXX call
    put (any: ANY)
        -- write class as string
        ensure
            successfull_write: last_written = any.out.count;
            need_flush_set: need_flush
    write_buffer (buf: STDC_BUFFER; offset, bytes: INTEGER)
        -- more safe version of write in case you have a
        -- STDC_BUFFER object
        -- Check last_written for number of bytes actually written,
        -- if you use asynchronous writing.
        require
            dont_write_garbage: buf.is_valid_index(offset + bytes - 1)
        ensure
            successfull_write: last_written = bytes;
            need_flush_set: need_flush
    write_boolean (b: BOOLEAN)
        -- write a boolean in Standard C %f format
        ensure
            successfull_write: last_written = b.out.count;
            need_flush_set: need_flush
    write_character (c: CHARACTER)
        -- write a single character
        ensure
            successfull_write: last_written = 1;
```

```
            need_flush_set: need_flush
        write_double (d: DOUBLE)
            -- write a double in Standard C %f format
            ensure
                need_flush_set: need_flush
        write_integer (i: INTEGER)
            -- write an integer in Standard C %d format
            ensure
                need_flush_set: need_flush
        write_real (r: REAL)
            -- write a real in Standard C %f format
            ensure
                need_flush_set: need_flush
        write_string (s: STRING)
            -- write a string
            ensure
                successfull_write: (s = Void implies last_written = 0) or else s /= Void implies last_written = s.count
                need_flush_set: s /= Void implies need_flush
        puts (s: STRING)
            -- write a string
            ensure
                successfull_write: (s = Void implies last_written = 0) or else s /= Void implies last_written = s.count
                need_flush_set: s /= Void implies need_flush
        put_string (s: STRING)
            -- write a string
            ensure
                successfull_write: (s = Void implies last_written = 0) or else s /= Void implies last_written = s.count
                need_flush_set: s /= Void implies need_flush
    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
            ensure
                eof_not_set: not eof
    feature(s) from STDC_FILE
        -- other
        clear_error
            -- Clears end-of-file and error indicators for a stream
            ensure
                eof_not_set: not eof
    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
feature(s) from STDC_FILE
    -- is mode binary or text
    is_binary_mode_specification (a_mode: STRING): BOOLEAN
        -- True if last character of a_mode = b
    is_text_mode_specification (a_mode: STRING): BOOLEAN
        -- True if last character of a_mode = t
feature(s) from STDC_TEXT_FILE
    -- text specific routines
    chop_last_string
        -- remove newline character (if any) from last_string
        -- only when the last characters from last_string are %N or
        -- %R%N, theyre removed.
    put_nl
        -- write a single newline
    put_newline
        -- write a single newline
    write_nl
        -- write a single newline
    write_newline
        -- write a single newline
invariant
    accessing_real_singleton: security_is_real_singleton;
    last_string_valid: last_string /= Void;
    gets_buf_valid: gets_buf /= Void;
end of STDC_TEXT_FILE
```

B.7 STDC_FILE_SYSTEM

```
class interface STDC_FILE_SYSTEM
feature(s) from STDC_FILE_SYSTEM
    -- path names
    expand_path (a_path: STRING): STDC_PATH
        -- returns a new path
feature(s) from STDC_FILE_SYSTEM
    -- rename files/directories, remove files/directories
    remove_file (a_path: STRING)
        -- Removes a file from a directory.
        -- For Standard C, its implementation defined what
        -- remove_file does if file is opened by some process
        -- (remove_file fails on Windows for example).
        -- doesnt remove a directory.
        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 attemting 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 attemting to open a_path file read-only
        require
            valid_path: a_path /= Void and then not a_path.is_empty
invariant
    accessing_real_singleton: security_is_real_singleton;
end of STDC_FILE_SYSTEM
```

B.8 STDC_SIGNAL

```
class interface STDC_SIGNAL
creation
    make (a_value: INTEGER)
        require
            valid_signal: a_value >= 1
feature(s) from STDC_SIGNAL
    -- creation
    make (a_value: INTEGER)
        require
            valid_signal: a_value >= 1
feature(s) from STDC_SIGNAL
    -- set signal properties, make effective with apply
    apply
        -- make changes effective
    set_default_action
        -- install signal-specific default action
    set_ignore_action
        -- ignore signal
        require
            ignorable: is_ignorable
    set_handler (a_handler: STDC_SIGNAL_HANDLER)
        -- install ones own signal handler
feature(s) from STDC_SIGNAL
    -- signal functions
    raise
        -- raise the signal
feature(s) from STDC_SIGNAL
    -- signal state
    is_ignorable: BOOLEAN
        -- All signals Standard C knows about are ignorable...
    value: INTEGER
        -- the signal
invariant
    accessing_real_singleton: signal_switch_is_real_singleton;
    accessing_real_singleton: security_is_real_singleton;
    valid_signal_value: value >= 1;
end of STDC_SIGNAL
```

B.9 STDC_SIGNAL_HANDLER

deferred class *interface STDC_SIGNAL_HANDLER*
invariant
accessing_real_singleton: *signal_switch_is_real_singleton*;
end *of* **deferred** *STDC_SIGNAL_HANDLER*

B.10 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
feature(s) from STDC_SYSTEM
    -- endianess
    is_big_endian: BOOLEAN
        -- True if this is a big endian architecture
    is_little_endian: BOOLEAN
        -- True if this is a little endian architecture
invariant
    accessing_real_singleton: security_is_real_singleton;
end of STDC_SYSTEM
```

B.11 STDC_TIME

```
class interface STDC_TIME
creation
    make_date (a_year, a_month, a_day: INTEGER)
        -- create a time according to this day, time 00:00:00
        -- date is assumed to be local date
    make_date_time (a_year, a_month, a_day, a_hour, a_minute, a_second: INTEGER)
        -- date is assumed to be local date
        -- we assume daylight saving time setting in effect is
        -- available from system
    make_from_now
        -- make value equal to current unix time
        -- afterwards call to_local or to_utc to turn individual
        -- fields in local time or in utc time.
    make_time (a_hour, a_minute, a_second: INTEGER)
        -- we assume daylight saving time setting in effect is
        -- available from system
        -- day will be January 1, 1970
    make_from_unix_time (a_value: INTEGER)
feature(s) from STDC_TIME
    -- creation
    make_date (a_year, a_month, a_day: INTEGER)
        -- create a time according to this day, time 00:00:00
        -- date is assumed to be local date
    make_date_time (a_year, a_month, a_day, a_hour, a_minute, a_second: INTEGER)
        -- date is assumed to be local date
        -- we assume daylight saving time setting in effect is
        -- available from system
    make_from_now
        -- make value equal to current unix time
        -- afterwards call to_local or to_utc to turn individual
        -- fields in local time or in utc time.
    make_time (a_hour, a_minute, a_second: INTEGER)
        -- we assume daylight saving time setting in effect is
        -- available from system
        -- day will be January 1, 1970
    make_from_unix_time (a_value: INTEGER)
feature(s) from STDC_TIME
    -- make individual time fields valid
    to_local
        -- switch time fields to local time
    to_utc
        -- switch time fields to utc time
feature(s) from STDC_TIME
    -- manually set individual time fields
    set_date (a_year, a_month, a_day: INTEGER)
```

```
        -- set date part, time remains unchanged
    set_date_time (a_year, a_month, a_day, a_hour, a_minute, a_second: INTEGER)
        -- we assume daylight saving time setting in effect (or not)
        -- has been set
        require
            valid_year: a_year >= minimum_year -- valid date/time: dont try to pass invalid values
    set_dst_to_current
        -- let system figure out if daylight saving time is in effect
    set_dst_to_none
        -- daylight saving time is not in effect
    set_time (a_hour, a_minute, a_second: INTEGER)
        -- set time part, date remains unchanged
feature(s) from STDC_TIME
    -- individual time fields, need call to to_local or to_utc
    year: INTEGER
    month: INTEGER
    day: INTEGER
        -- Day of the month
    weekday: INTEGER
        -- Days since Sunday
    day_of_year: INTEGER
        -- Days since January 1st
    hour: INTEGER
    minute: INTEGER
    second: INTEGER
feature(s) from STDC_TIME
    -- time as string
    short_weekday_name: STRING
        -- Abbreviated weekday name
    weekday_name: STRING
        -- Full weekday name
    short_month_name: STRING
        -- Abbreviated month name
    month_name: STRING
        -- Full month name
    format (format_str: STRING): STRING
    default_format: STRING
        -- Returns a string of the form "Mon Apr 17 21:49:20 2000"
    local_date_string: STRING
        -- Return date part in format local to current country
    local_time_string: STRING
        -- Return time part in format local to current country
feature(s) from STDC_TIME
    -- date calculations
    is_equal (other: like Current): BOOLEAN
        -- Is other attached to an object considered equal to
        -- current object ?
```

```
        require
            other_not_void: other /= Void
        ensure
            consistent: standard_is_equal(other) implies Result;
            symmetric: Result implies other.is_equal(Current);
            trichotomy: Result = (not (Current < other) and not (other < Current))
    infix "-" (other: like Current): like Current
        -- Creates a new time which is the difference between
        -- Current and Other
    infix "<" (other: like Current): BOOLEAN
        -- Is current object less than other?
        require
            other_exists: other /= Void
        ensure
            asymmetric: Result implies not (other < Current)
feature(s) from STDC_TIME
    -- state
    hash_code: INTEGER
        -- The hash-code value of Current.
        ensure
            good_hash_value: Result >= 0
    value: INTEGER
        -- time in seconds from January 1, 1970
        -- perhaps since 1980 for Windows systems
feature(s) from STDC_TIME
    -- non POSIX, but Gates specific stuff
    minimum_year: INTEGER
        -- returns the minimum year for the current platform
        -- for POSIX is 1970, for Windows is 1980
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_tm_struct: tm /= Void;
end of STDC_TIME
```

In this chapter:

C Short listing of abstract classes

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

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

C.1 ABSTRACT_CURRENT_PROCESS

```
deferred class interface ABSTRACT_CURRENT_PROCESS
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from STDC_CURRENT_PROCESS
    -- my stdandard input/output/error
    stdin: STDC_TEXT_FILE
    stdout: STDC_TEXT_FILE
    stderr: STDC_TEXT_FILE
feature(s) from ABSTRACT_PROCESS
    -- process properties
    pid: INTEGER
        -- the process identifier
        require
            valid_pid: is_pid_valid
        ensure
            valid_pid: Result > 0
    is_pid_valid: BOOLEAN
        -- current process id is always valid
feature(s) from ABSTRACT_PROCESS
    -- signal this process
    terminate
        -- attempt to gracefully terminate this process
```

```
        require
            valid_pid: is_pid_valid
    feature(s) from ABSTRACT_CURRENT_PROCESS
        -- every process also has standard file descriptors which might not be compatible with stdin/stdout/stderr (Wi
        fd_stdin: ABSTRACT_FILE_DESCRIPTOR
        fd_stdout: ABSTRACT_FILE_DESCRIPTOR
        fd_stderr: ABSTRACT_FILE_DESCRIPTOR
    invariant
        accessing_real_singleton: security_is_real_singleton;
    end of deferred ABSTRACT_CURRENT_PROCESS
```

C.2 ABSTRACT_EXEC_PROCESS

```
deferred class interface ABSTRACT_EXEC_PROCESS
    -- inheriting from this gives clash in POSIX_FORK_ROOT
    -- I didnt need a correct inheritance tree at this moment
    -- anyway, so I left it disabled.
    -- This means you have to inherit from ABSTRACT_CHILD_PROCESS!
feature(s) from ABSTRACT_EXEC_PROCESS
    -- creation
    make (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_input (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_output (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_io (a_program: STRING; a_arguments: ARRAY[STRING])
        -- Why not use threedirectional i/o, because youre getting
        -- yourself in great, great trouble anyway.
        -- A bit of advice: call stdin.close before starting to call
        -- stdout.read_string and such...
    make_capture_all (a_program: STRING; a_arguments: ARRAY[STRING])
        -- Threedirectional i/o is a great way to get yourself in trouble.
feature(s) from ABSTRACT_EXEC_PROCESS
    -- (re)set arguments
    set_arguments (a_arguments: ARRAY[STRING])
feature(s) from ABSTRACT_EXEC_PROCESS
    -- i/o capturing
    capture_input: BOOLEAN
        -- is input captured on execute?
    capture_output: BOOLEAN
        -- is output captured on execute?
    capture_error: BOOLEAN
        -- is error captured on execute?
    set_capture_input (on: BOOLEAN)
    set_capture_output (on: BOOLEAN)
    set_capture_error (on: BOOLEAN)
    fd_stdin: ABSTRACT_FILE_DESCRIPTOR
    fd_stdout: ABSTRACT_FILE_DESCRIPTOR
    fd_stderr: ABSTRACT_FILE_DESCRIPTOR
feature(s) from ABSTRACT_EXEC_PROCESS
    -- execute
    execute
        -- Executes program_name
        -- dont forget to wait for this process to terminate
        require
            not_already_started: is_terminated
feature(s) from ABSTRACT_EXEC_PROCESS
    -- actions that parent may execute
    wait_for (suspend: BOOLEAN)
        require
```

```
            pid_refers_to_child: is_pid_valid;
            not_terminated: not is_terminated
        ensure
            stdin_closed: is_terminated implies fd_stdin = Void or else fd_stdin.is_closed;
            stdout_closed: is_terminated implies fd_stdout = Void or else fd_stdout.is_closed;
            stderr_closed: is_terminated implies fd_stderr = Void or else fd_stderr.is_closed
feature(s) from ABSTRACT_EXEC_PROCESS
    -- accessible state
    program_name: STDC_PATH
        -- program to execute
    arguments: ARRAY[STRING]
        -- arguments to pass to program
invariant
    accessing_real_singleton: security_is_real_singleton;
end of deferred ABSTRACT_EXEC_PROCESS
```

C.3 ABSTRACT_FILE_DESCRIPTOR

```
deferred class interface ABSTRACT_FILE_DESCRIPTOR
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- creation
    open (a_path: STRING; a_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_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. Make sure you have th O_CREAT flag in flags
        -- if you really want to create something!
        require
            closed: is_closed
```

```
feature(s) from ABSTRACT_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;
            not_owner: not is_owner
    make_as_duplicate (another: ABSTRACT_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;
            own_descriptor: is_owner
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- close
    close
        -- we always describe an existing object, however user probably wants
        -- to have control about closing a file.
        require
            opened: is_open
        ensure
            closed: is_closed
    unattach
        -- unbind from the current file descriptor
        ensure
            closed: not is_open
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- change ownership of the descriptor. Can help to influence subtile garbage collector problems
    make_owner
        -- this file descriptor will (start to) own its descriptor
        ensure
            owner: is_owner
    unown
        -- when a stream is opened on a file descriptor the file
        -- descriptor itself should not close itself, the stream
        -- will close it
        ensure
            not_owner: not is_owner
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- raw read and write
    last_blocked: BOOLEAN
        -- True if last read or write call would be blocked
    last_read: INTEGER
```

```
        -- how many bytes were read by last call to read
        -- -1 implies last_blocked
    last_written: INTEGER
        -- how many bytes were written by last call to write
        -- -1 implies last_blocked
    read_loop_disabled: BOOLEAN
        -- for certain applications rereading isnt very smart,
        -- i.e. it will block the application. In such cases read just what
        -- is available, do not attempt to read more.  Such file
        -- descriptors do not return an EOF, but just block.
        -- A typical example is a character special file.
    read (buf: POINTER; offset, nbytes: INTEGER)
        -- read data into buf at offset for nbytes bytes.
        -- number of bytes actually read are available in last_read
        require
            can_read: is_open;
            valid_buf: buf /= default_pointer;
            valid_size: nbytes >= 0 -- nbytes < SSIZE_MAX (POSIX value?)
        ensure
            read_no_more_than_requested: last_read <= nbytes
    write (buf: POINTER; offset, nbytes: INTEGER)
        -- write given data from buf at offset, for nbytes bytes.
        require
            can_write: is_open;
            valid_buf: buf /= default_pointer;
            valid_size: nbytes >= 0
        ensure
            wrote_no_more_than_requested: last_written <= nbytes
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- safer read/write
    read_buffer (buf: STDC_BUFFER; offset, nbytes: INTEGER)
        -- more safe version of read in case you have a
        -- STDC_BUFFER object
        require
            dont_overflow: buf.capacity >= offset + nbytes
    write_buffer (buf: STDC_BUFFER; offset, bytes: INTEGER)
        -- more safe version of write in case you have a
        -- STDC_BUFFER object
        require
            dont_write_garbage: buf.is_valid_index(offset + bytes - 1)
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- buffered input/output, line reading instead of block reading
    last_character: CHARACTER
    last_string: STRING
        -- last read string (includes %N), see STRING_HELPER.chop
    put (a: ANY)
        -- write any Eiffel object as string
```

```
    read_character
        -- set last_character.
    read_string (a_size: INTEGER)
        -- implements line reading on top of read. Sets last_string
        -- which includes the new line character if any.
        require
            valid_size: a_size >= 0
    write_character (c: CHARACTER)
    write_string (s: STRING)
    puts (s: STRING)
    put_string (s: STRING)
feature(s) from ABSTRACT_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 ABSTRACT_FILE_DESCRIPTOR
    -- queries
    eof: BOOLEAN
        -- True if end-of-file reached.
        -- Currently Im unsure if detection is reliable
    isatty: BOOLEAN
        -- return true if handle associated with character device
    is_attached_to_terminal: BOOLEAN
        -- return true if handle associated with character device
    is_blocking_io: BOOLEAN
        -- True if blocking i/o enabled (default)
    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
    is_owner: BOOLEAN
        -- does this file descriptor own its descriptor?
        -- only when it owns the descriptor it will automatically close it
    status: ABSTRACT_STATUS
    value: INTEGER
```

```
        -- return the value of the file descriptor
        require
            valid_file_descriptor: is_open
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- accessible state
    path: STRING
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_internal_file_descriptor: fd >= - 1;
    valid_open: is_open implies fd >= 0;
    valid_close: not is_open implies fd = - 1;
    valid_status: is_closed implies my_status = Void;
end of deferred ABSTRACT_FILE_DESCRIPTOR
```

C.4 ABSTRACT_FILE_SYSTEM

```
deferred class interface ABSTRACT_FILE_SYSTEM
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from STDC_FILE_SYSTEM
    -- path names
    expand_path (a_path: STRING): STDC_PATH
        -- returns a new path
feature(s) from STDC_FILE_SYSTEM
    -- rename files/directories, remove files/directories
    remove_file (a_path: STRING)
        -- Removes a file from a directory.
        -- For Standard C, its implementation defined what
        -- remove_file does if file is opened by some process
        -- (remove_file fails on Windows for example).
        -- doesnt remove a directory.
        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
        -- 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
feature(s) from ABSTRACT_FILE_SYSTEM
    -- directory access
```

```
    change_directory (a_directory: STRING)
        -- Changes the current working directory
feature(s) from ABSTRACT_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 an empty directory, see also force_remove_directory
        require
            valid_directory_name: a_directory /= Void and then not a_directory.is_empty;
            directory_is_empty: true
    rmdir (a_directory: STRING)
        -- Removes an empty directory, see also force_remove_directory
        require
            valid_directory_name: a_directory /= Void and then not a_directory.is_empty;
            directory_is_empty: true
    force_remove_directory (a_directory: STRING)
        -- Removes a directory, even when not empty
        require
            valid_path: a_directory /= Void and then not a_directory.is_empty
feature(s) from ABSTRACT_FILE_SYSTEM
    -- file statistics
    status (a_path: STRING): ABSTRACT_STATUS
        -- Gets information about a file
        require
            valid_path: a_path /= Void and then not a_path.is_empty;
            existing_file: is_existing(a_path)
        ensure
            status_returned: Result /= Void
feature(s) from ABSTRACT_FILE_SYSTEM
    -- directory browsing
    browse_directory (a_path: STRING): ABSTRACT_DIRECTORY
        -- Gets information about a directory
        require
            valid_path: a_path /= Void and then not a_path.is_empty;
            path_is_directory: security.error_handling.exceptions_enabled and then status(a_path).is_directory
        ensure
```

```
            directory_returned: Result /= Void
feature(s) from ABSTRACT_FILE_SYSTEM
    -- accessibility of files
    last_access_result: INTEGER
        -- value of last access test
    is_accessible (a_path: STRING; a_mode: INTEGER): BOOLEAN
        -- Tests for file accessibility
    access (a_path: STRING; a_mode: INTEGER): BOOLEAN
        -- Tests for file accessibility
    is_directory (a_path: STRING): BOOLEAN
        -- return True if a_path exists and if it is a directory
    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_writable (a_path: STRING): BOOLEAN
        -- tests if file is writable by this program
        -- uses real user ID and real group ID instead of effective ones
feature(s) from ABSTRACT_FILE_SYSTEM
    -- various
    is_case_sensitive: BOOLEAN
        -- is file system case sensitive or not?
        -- This query is dedicated to jwz
    path_separator: CHARACTER
        -- What is the path separator?
feature(s) from ABSTRACT_FILE_SYSTEM
    -- file system properties
    temporary_directory: STRING
        -- returns temporary directory.
        ensure
            directory_returned: Result /= Void;
            directory_exists: is_directory(Result);
            directory_is_writable: is_modifiable(Result);
            last_char_not_separator: Result.item(Result.count) /= path_separator
invariant
    accessing_real_singleton: security_is_real_singleton;
end of deferred ABSTRACT_FILE_SYSTEM
```

C.5 ABSTRACT_PIPE

```
deferred class interface ABSTRACT_PIPE
feature(s) from ABSTRACT_PIPE
    -- creation
    make
feature(s) from ABSTRACT_PIPE
    -- pipe operations
    close
feature(s) from ABSTRACT_PIPE
    -- the pipe
    fdout: ABSTRACT_FILE_DESCRIPTOR
    fdin: ABSTRACT_FILE_DESCRIPTOR
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_pipe: fdin /= Void and fdout /= Void;
end of deferred ABSTRACT_PIPE
```

C.6 ABSTRACT_STATUS

```
deferred class interface ABSTRACT_STATUS
feature(s) from ABSTRACT_STATUS
    refresh
        -- refresh the cached information
feature(s) from ABSTRACT_STATUS
    -- stat members
    atime: INTEGER
        -- Unix time of last access
feature(s) from ABSTRACT_STATUS
    -- stat members
    access_time: INTEGER
        -- Unix time of last access
    ctime: INTEGER
        -- Unix time of last status change, for example changing permission
        -- bits
    change_time: INTEGER
        -- Unix time of last status change, for example changing permission
        -- bits
    device_number: INTEGER
        -- ID of device containing the file
        -- Windows: Drive number of the disk containing the file
    is_character_special: BOOLEAN
        -- True if character-special file
    is_directory: BOOLEAN
    is_fifo: BOOLEAN
    is_regular_file: BOOLEAN
    mtime: INTEGER
        -- Unix time of last data modification
    modification_time: INTEGER
        -- Unix time of last data modification
    nlink: INTEGER
    number_of_hard_links: INTEGER
    size: INTEGER
        -- size of file in bytes
feature(s) from ABSTRACT_STATUS
    -- direct access to the individual stat fields, not recommended
    unix_mode: INTEGER
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_stat: stat /= Void;
end of deferred ABSTRACT_STATUS
```

In this chapter:

D Short (flat) listing of POSIX classes

D.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
    raw_pointer: POINTER
        -- Location for read or written data, usually buffer is a
        -- better idea.
    count: INTEGER
        -- number of bytes to read/write
    offset: INTEGER
        -- file offset
feature(s) from POSIX_ASYNC_IO_REQUEST
    -- set request properties
    set_buffer (a_buffer: STDC_BUFFER)
        -- set memory location to read/write from.
        require
            nothing_pending: not is_pending;
            valid_buffer: a_buffer /= Void
        ensure
            buffer_set: buffer = a_buffer;
            raw_pointer_set: raw_pointer = a_buffer.ptr
    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
    set_raw_pointer (a_pointer: POINTER)
        -- set memory location to read/write from. Make sure you have
        -- called set_count first!
        require
            nothing_pending: not is_pending;
            need_count: count > 0;
            valid_pointer: a_pointer /= default_pointer
        ensure
            buffer_set: buffer.ptr = a_pointer;
            raw_pointer_set: raw_pointer = a_pointer
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;
            has_buffer: raw_pointer /= default_pointer and buffer /= Void;
            has_capacity: buffer.capacity >= count
    write
        -- execute async write request
        require
            is_open: fd.is_open;
            nothing_pending: not is_pending;
            has_buffer: raw_pointer /= default_pointer and buffer /= Void;
            has_capacity: buffer.capacity >= count
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!
        require
            nothing_pending: not is_pending
    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
    buffer: STDC_BUFFER
        -- buffer where data that is being read/write comes from,
        -- unless set_pointer has been called
    fd: POSIX_FILE_DESCRIPTOR
    is_pending: BOOLEAN
        -- 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
    accessing_real_singleton: security_is_real_singleton;
    valid_aiocb: aiocb /= Void;
    synced_buffer_and_raw_pointer: buffer /= Void implies buffer.ptr = raw_pointer;
end of POSIX_ASYNC_IO_REQUEST
```

D.2 POSIX_BASE

class *interface* *POSIX_BASE*
invariant
accessing_real_singleton: *security_is_real_singleton*;
end *of* *POSIX_BASE*

D.3 POSIX_CHILD_PROCESS

```
deferred class interface POSIX_CHILD_PROCESS
feature(s) from POSIX_CHILD_PROCESS
    -- childs 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.
        -- If suspend is False, check the running property to see
        -- if this child is really terminated.
        require
            pid_refers_to_child: is_pid_valid;
            not_terminated: not is_terminated
invariant
    accessing_real_singleton: security_is_real_singleton;
end of deferred POSIX_CHILD_PROCESS
```

D.4 POSIX_CONSTANTS

class *interface* *POSIX_CONSTANTS*
feature(*s*) **from** *STDC_CONSTANTS*
-- error codes
EDOM: *INTEGER*
-- Math argument out of domain of function
ERANGE: *INTEGER*
-- Math result not representable
feature(*s*) **from** *STDC_CONSTANTS*
-- standard streams
stream_stdin: *POINTER*
stream_stdout: *POINTER*
stream_stderr: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
-- characters
const_EOF: *INTEGER*
-- signals EOF
feature(*s*) **from** *STDC_CONSTANTS*
-- buffering
IOFBF: *INTEGER*
-- full buffering
IOLBF: *INTEGER*
-- line buffering
IONBF: *INTEGER*
-- no buffering
feature(*s*) **from** *STDC_CONSTANTS*
-- file positioning
SEEK_SET: *INTEGER*
SEEK_CUR: *INTEGER*
SEEK_END: *INTEGER*
feature(*s*) **from** *STDC_CONSTANTS*
-- Signal related constants
SIG_DFL: *POINTER*
SIG_ERR: *POINTER*
SIG_IGN: *POINTER*
feature(*s*) **from** *STDC_CONSTANTS*
-- Signals
SIGABRT: *INTEGER*
SIGFPE: *INTEGER*
-- erroneous arithmetic operation, such as zero divide or an
-- operation resulting in overflow
SIGILL: *INTEGER*
-- illegal instruction
SIGINT: *INTEGER*
-- receipt of an interactive attention signal
SIGSEGV: *INTEGER*

-- invalid access to storage
SIGTERM: *INTEGER*
feature(*s*) **from** *STDC_CONSTANTS*
-- random numbers
RAND_MAX: *INTEGER*
-- maximum value returned by the *random* function
feature(*s*) **from** *STDC_CONSTANTS*
-- category constants
LC_CTYPE: *INTEGER*
LC_NUMERIC: *INTEGER*
LC_TIME: *INTEGER*
LC_COLLATE: *INTEGER*
LC_MONETARY: *INTEGER*
LC_ALL: *INTEGER*
feature(*s*) **from** *STDC_CONSTANTS*
-- various
CLOCKS_PER_SEC: *INTEGER*
feature(*s*) **from** *STDC_CONSTANTS*
-- exit codes
EXIT_FAILURE: *INTEGER*
-- exit status when something has gone wrong
EXIT_SUCCESS: *INTEGER*
-- exit status upon success
feature(*s*) **from** *POSIX_CONSTANTS*
-- error codes
EAGAIN: *INTEGER*
EBADF: *INTEGER*
EEXIST: *INTEGER*
EINPROGRESS: *INTEGER*
EINTR: *INTEGER*
ENOENT: *INTEGER*
-- A file or directory does not exist
ENOSPC: *INTEGER*
-- There is no free space remaining on the device
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 open symbolic constants
O_APPEND: *INTEGER*
-- Set the file offset to the end-of-file prior to each write
O_CREAT: *INTEGER*
-- If the file does not exist, allow it to be created. This
-- flag indicates that the mode argument is present in the

-- call to open.
O_DSYNC: *INTEGER*
-- Write according to synchronized i/o data integrity completion
O_EXCL: *INTEGER*
-- Open fails if the file already exists
O_EXCLUSIVE: *INTEGER*
-- Open fails if the file already exists
O_NOCTTY: *INTEGER*
-- prevents terminal from becoming the controlling terminal
-- for this process
O_NONBLOCK: *INTEGER*
-- Do not wait for device or file to be ready or available
O_RDONLY: *INTEGER*
-- Open for reading only
O_RDWR: *INTEGER*
-- Open fo reading and writing
O_RSYNC: *INTEGER*
-- Synchronized read i/o operations
O_SYNC: *INTEGER*
-- Write according to synchronized i/o file integrity completion
O_TRUNC: *INTEGER*
-- Use only on ordinary files opened for writing. It causes
-- the file to be truncated to zero length.
O_WRONLY: *INTEGER*
-- Open for writing only
feature(*s*) **from** *POSIX_CONSTANTS*
-- posix permission symbolic constants
S_IRUSR: *INTEGER*
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 accessibility constants
F_OK: *INTEGER*
R_OK: *INTEGER*

W_OK: *INTEGER*
X_OK: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- Posix signal constants
SA_NOCLDSTOP: *INTEGER*
SIGHUP: *INTEGER*
-- hangup detected on controlling terminal or death of
-- controlling process
SIGNAL_HANGUP: *INTEGER*
-- hangup detected on controlling terminal or death of
-- controlling process
SIGALRM: *INTEGER*
-- Timeout signal, such as initiated by the alarm() function
-- or see POSIX_TIMED_COMMAND
SIGNAL_ALARM: *INTEGER*
-- Timeout signal, such as initiated by the alarm() function
-- or see POSIX_TIMED_COMMAND
SIGCHLD: *INTEGER*
-- Child process terminated or stopped
SIGNAL_CHILD: *INTEGER*
-- Child process terminated or stopped
SIGKILL: *INTEGER*
-- Termination signal (cannot be caught or ignored)
SIGNAL_KILL: *INTEGER*
-- Termination signal (cannot be caught or ignored)
SIGPIPE: *INTEGER*
-- Write on a pipe with no readers
SIGNAL_PIPE: *INTEGER*
-- Write on a pipe with no readers
SIGQUIT: *INTEGER*
-- Interactive termination signal
SIGNAL_QUIT: *INTEGER*
-- Interactive termination signal
SIGCONT: *INTEGER*
-- Continue if stopped
SIGNAL_CONTINUE: *INTEGER*
-- Continue if stopped
SIGSTOP: *INTEGER*
-- Stop signal, cannot be caught or ignored
SIGNAL_STOP: *INTEGER*
-- Stop signal, cannot be caught or ignored
SIGTSTP: *INTEGER*
-- Interactive stop signal
SIGNAL_INTERACTIVE_STOP: *INTEGER*
-- Interactive stop signal
SIGTTIN: *INTEGER*
-- Read from control terminal attempted by a member of a

-- background process group
SIGNAL_TERMINAL_IN: *INTEGER*
-- Read from control terminal attempted by a member of a
-- background process group
SIGTTOU: *INTEGER*
-- Write to control terminal attempted by a member of a
-- background process group
SIGNAL_TERMINAL_OUT: *INTEGER*
-- Write to control terminal attempted by a member of a
-- background process group
feature(*s*) **from** *POSIX_CONSTANTS*
-- sigprocmask how values
SIG_BLOCK: *INTEGER*
SIG_UNBLOCK: *INTEGER*
SIG_SETMASK: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- Posix pathconf constants
PC_NAME_MAX: *INTEGER*
-- The maximum length of a filename for this directory
feature(*s*) **from** *POSIX_CONSTANTS*
-- terminal i/o local mode flags
ISIG: *INTEGER*
ICANON: *INTEGER*
ECHO: *INTEGER*
-- If set, input characters are echoed back to the terminal
ECHOE: *INTEGER*
ECHOK: *INTEGER*
ECHONL: *INTEGER*
NOFLSH: *INTEGER*
TOSTOP: *INTEGER*
IEXTEN: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- set terminal settings options
Tcsanow: *INTEGER*
Tcsadrain: *INTEGER*
Tcsaflush: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- semaphore constants
SEM_VALUE_MAX: *INTEGER*
-- Valid Maximum initial value for a semaphore
feature(*s*) **from** *POSIX_CONSTANTS*
-- terminal baud rates
B0: *INTEGER*
B50: *INTEGER*
B75: *INTEGER*
B110: *INTEGER*
B134: *INTEGER*

B150: *INTEGER*
B200: *INTEGER*
B300: *INTEGER*
B600: *INTEGER*
B1200: *INTEGER*
B1800: *INTEGER*
B2400: *INTEGER*
B4800: *INTEGER*
B9600: *INTEGER*
B19200: *INTEGER*
B38400: *INTEGER*
B57600: *INTEGER*
B115200: *INTEGER*
B230400: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- terminal i/o control mode constants
CSIZE: *INTEGER*
CS5: *INTEGER*
CS6: *INTEGER*
CS7: *INTEGER*
CS8: *INTEGER*
CSTOPB: *INTEGER*
CREAD: *INTEGER*
PARENB: *INTEGER*
PARODD: *INTEGER*
HUPCL: *INTEGER*
CLOCAL: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- terminal i/o input control flags
IGNBRK: *INTEGER*
BRKINT: *INTEGER*
IGNPAR: *INTEGER*
PARMRK: *INTEGER*
INPCK: *INTEGER*
ISTRIP: *INTEGER*
INLCR: *INTEGER*
IGNCR: *INTEGER*
ICRNL: *INTEGER*
IXON: *INTEGER*
IXOFF: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- category constants
LC_MESSAGES: *INTEGER*
feature(*s*) **from** *POSIX_CONSTANTS*
-- pathname variable values
MAX_INPUT: *INTEGER*
-- Minimum number of bytes for which space will be available

```
        -- in a terminal input queue; therefore, the maximum number
        -- of bytes a portable application may required to be typed
        -- as input before eading them
    NAME_MAX: INTEGER
        -- Maximum number of bytes in a file name
    PATH_MAX: INTEGER
        -- Maximum number of bytes in a pathname
    PIPE_BUF: INTEGER
        -- Maximum number of bytes that can be written atomically
        -- when writing to a pipe.
feature(s) from POSIX_CONSTANTS
    -- invariant values
    SSIZE_MAX: INTEGER
        -- The maximum value that can be stored in an object of type ssize_t
end of POSIX_CONSTANTS
```

D.5 POSIX_CURRENT_PROCESS

class *interface POSIX_CURRENT_PROCESS*
feature(*s*) **from** *STDC_CURRENT_PROCESS*
-- my stdandard input/output/error
stdin: *POSIX_TEXT_FILE*
stdout: *POSIX_TEXT_FILE*
stderr: *POSIX_TEXT_FILE*
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- process properties
pid: *INTEGER*
-- the process identifier
require
valid_pid: *is_pid_valid*
ensure
valid_pid: *Result > 0*
is_pid_valid: *BOOLEAN*
-- current process id is always valid
feature(*s*) **from** *ABSTRACT_CURRENT_PROCESS*
-- every process also has standard file descriptors which might not be compatible with stdin/stdout/stderr (Wi
fd_stdin: *POSIX_FILE_DESCRIPTOR*
fd_stdout: *POSIX_FILE_DESCRIPTOR*
fd_stderr: *POSIX_FILE_DESCRIPTOR*
feature(*s*) **from** *STDC_SECURITY_ACCESSOR*
-- the singleton, available to any because its used in preconditions
security: *STDC_SECURITY*
-- singleton entry point for security
ensure
has_security: *Result /= Void*
feature(*s*) **from** *STDC_BASE*
-- errno
errno: *STDC_ERRNO*
feature(*s*) **from** *ABSTRACT_PROCESS*
-- signal this process
terminate
-- attempt to gracefully terminate this process
require
valid_pid: *is_pid_valid*
feature(*s*) **from** *POSIX_PROCESS*
-- signal this process
kill (*a_signal_code*: *INTEGER*)
-- Send signal *signal_code* to the process
require
valid_pid: *is_pid_valid*;
valid_signal: *a_signal_code >= 0* -- not_terminated: not is_terminated
feature(*s*) **from** *POSIX_CURRENT_PROCESS*
-- POSIX locale specifics

```
    set_native_messages
        -- Select native language as the language in which messages
        -- are displayed
invariant
    accessing_real_singleton: security_is_real_singleton;
end of POSIX_CURRENT_PROCESS
```

D.6 POSIX_DAEMON

```
deferred class interface POSIX_DAEMON
feature(s) from POSIX_DAEMON
    -- daemon specific actions
    detach
        -- detach from command-line, not very useful if you want to
        -- spawn multiple daemons, but you can always pass daemons to
        -- the fork routine yourself.
    after_fork
        -- Code thanks to W. Richard Stevens
invariant
    accessing_real_singleton: security_is_real_singleton;
end of deferred POSIX_DAEMON
```

D.7 POSIX_DIRECTORY

```
class interface POSIX_DIRECTORY
creation
    make (a_directory_name: STRING)
feature(s) from POSIX_DIRECTORY
    max_filename_length: INTEGER
        -- maximum length of a file in this directory
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_directory_name: directory_name /= Void;
end of POSIX_DIRECTORY
```

D.8 POSIX_EXEC_PROCESS

```
class interface POSIX_EXEC_PROCESS
creation
    make (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_input (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_output (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_io (a_program: STRING; a_arguments: ARRAY[STRING])
        -- Why not use threedirectional i/o, because youre getting
        -- yourself in great, great trouble anyway.
        -- A bit of advice: call stdin.close before starting to call
        -- stdout.read_string and such...
    make_capture_all (a_program: STRING; a_arguments: ARRAY[STRING])
        -- Threedirectional i/o is a great way to get yourself in trouble.
feature(s) from STDC_CHILD_PROCESS
    -- termination info
    is_terminated: BOOLEAN
        -- True if process 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
feature(s) from ABSTRACT_CHILD_PROCESS
    -- actions that parent may execute
    wait_for (suspend: BOOLEAN)
        -- wait for this process to terminate. If suspend then we
        -- wait until the information about this process is available,
        -- else we return immediately.
        -- If suspend is False, check the running property to see
        -- if this child is really terminated.
        require
            pid_refers_to_child: is_pid_valid;
            not_terminated: not is_terminated
        ensure
            stdin_closed: is_terminated implies fd_stdin = Void or else fd_stdin.is_closed;
            stdout_closed: is_terminated implies fd_stdout = Void or else fd_stdout.is_closed;
            stderr_closed: is_terminated implies fd_stderr = Void or else fd_stderr.is_closed
feature(s) from STDC_CURRENT_PROCESS
    -- my stdandard input/output/error
    child_stdin: POSIX_TEXT_FILE
    child_stdout: POSIX_TEXT_FILE
    child_stderr: POSIX_TEXT_FILE
feature(s) from ABSTRACT_CURRENT_PROCESS
    -- process properties
    pid: INTEGER
```

```
        -- either the current process identifier or the childs
        require
            valid_pid: is_pid_valid
        ensure
            valid_pid: Result > 0
    is_pid_valid: BOOLEAN
        -- current process id is always valid
feature(s) from ABSTRACT_CURRENT_PROCESS
    -- every process also has standard file descriptors which might not be compatible with stdin/stdout/stderr (Wi
    child_fd_stin: POSIX_FILE_DESCRIPTOR
    child_fd_stdout: POSIX_FILE_DESCRIPTOR
    child_fd_sterr: POSIX_FILE_DESCRIPTOR
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from ABSTRACT_PROCESS
    -- signal this process
    terminate
        -- attempt to gracefully terminate this process
        require
            valid_pid: is_pid_valid
feature(s) from POSIX_PROCESS
    -- signal this process
    kill (a_signal_code: INTEGER)
        -- Send signal signal_code to the process
        require
            valid_pid: is_pid_valid;
            valid_signal: a_signal_code >= 0 -- not_terminated: not is_terminated
feature(s) from POSIX_CURRENT_PROCESS
    -- POSIX locale specifics
    set_native_messages
        -- Select native language as the language in which messages
        -- are displayed
feature(s) from ABSTRACT_EXEC_PROCESS
    -- creation
    make (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_input (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_output (a_program: STRING; a_arguments: ARRAY[STRING])
    make_capture_io (a_program: STRING; a_arguments: ARRAY[STRING])
        -- Why not use threedirectional i/o, because youre getting
        -- yourself in great, great trouble anyway.
```

```
        -- A bit of advice: call stdin.close before starting to call
        -- stdout.read_string and such...
    make_capture_all (a_program: STRING; a_arguments: ARRAY[STRING])
        -- Threedirectional i/o is a great way to get yourself in trouble.
feature(s) from ABSTRACT_EXEC_PROCESS
    -- (re)set arguments
    set_arguments (a_arguments: ARRAY[STRING])
feature(s) from ABSTRACT_EXEC_PROCESS
    -- i/o capturing
    capture_input: BOOLEAN
        -- is input captured on execute?
    capture_output: BOOLEAN
        -- is output captured on execute?
    capture_error: BOOLEAN
        -- is error captured on execute?
    set_capture_input (on: BOOLEAN)
    set_capture_output (on: BOOLEAN)
    set_capture_error (on: BOOLEAN)
    fd_stdin: POSIX_FILE_DESCRIPTOR
    fd_stdout: POSIX_FILE_DESCRIPTOR
    fd_stderr: POSIX_FILE_DESCRIPTOR
feature(s) from ABSTRACT_EXEC_PROCESS
    -- execute
    execute
        -- Executes program_name
        -- dont forget to wait for this process to terminate
        require
            not_already_started: is_terminated
feature(s) from ABSTRACT_EXEC_PROCESS
    -- accessible state
    program_name: STDC_PATH
        -- program to execute
    arguments: ARRAY[STRING]
        -- arguments to pass to program
feature(s) from POSIX_FORK_ROOT
    -- process properties
    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
    after_fork
        -- chance for code to do something before the main execute
        -- mainly here for POSIX_DAEMON
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
    is_signalled: 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_signalled
feature(s) from EPX_EXEC_PROCESS
    -- i/o capturing
    stdin: POSIX_TEXT_FILE
    stdout: POSIX_TEXT_FILE
    stderr: POSIX_TEXT_FILE
invariant
    accessing_real_singleton: security_is_real_singleton;
end of POSIX_EXEC_PROCESS
```

D.9 POSIX_FILE

```
deferred class interface POSIX_FILE
feature(s) from POSIX_FILE
    -- special makes
    make_from_file_descriptor (a_file_descriptor: ABSTRACT_FILE_DESCRIPTOR; a_mode: STRING)
        -- open a stream from a given file descriptor
        -- The stream will become remains leading so when the file
        -- descriptor is closed, it will not close, you have to close
        -- the strema to close the file descriptor.
        require
            is_closed: not is_open;
            valid_fildes: a_file_descriptor /= Void and then a_file_descriptor.is_open -- valid_mode: a_mode is a v
        ensure
            opened: is_open;
            stream_is_leading: not a_file_descriptor.is_owner
invariant
    accessing_real_singleton: security_is_real_singleton;
    last_string_valid: last_string /= Void;
    gets_buf_valid: gets_buf /= Void;
end of deferred POSIX_FILE
```

D.10 POSIX_FILE_DESCRIPTOR

```
class interface POSIX_FILE_DESCRIPTOR
creation
    open (a_path: STRING; a_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_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. Make sure you have th O_CREAT flag in flags
        -- if you really want to create something!
        require
            closed: is_closed
    make_as_duplicate (another: ABSTRACT_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;
            own_descriptor: is_owner
    make_from_file (file: STDC_FILE)
        -- Create file descriptor from given stream
        -- The stream is leading, so this file descriptor will
```

```
        -- never close itself, unless it is made an owner.
        require
            closed: is_closed;
            valid_file: file /= Void and then file.is_open
        ensure
            open: is_open;
            not_owner: not is_owner
    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;
            not_owner: not is_owner
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- creation
    open (a_path: STRING; a_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_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. Make sure you have th O_CREAT flag in flags
        -- if you really want to create something!
        require
            closed: is_closed
feature(s) from ABSTRACT_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;
            not_owner: not is_owner
    make_as_duplicate (another: ABSTRACT_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;
            own_descriptor: is_owner
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- close
    close
        -- we always describe an existing object, however user probably wants
        -- to have control about closing a file.
        require
            opened: is_open
        ensure
            closed: is_closed
    unattach
        -- unbind from the current file descriptor
        ensure
            closed: not is_open
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- change ownership of the descriptor. Can help to influence subtile garbage collector problems
    make_owner
        -- this file descriptor will (start to) own its descriptor
        ensure
```

```
            owner: is_owner
    unown
        -- when a stream is opened on a file descriptor the file
        -- descriptor itself should not close itself, the stream
        -- will close it
        ensure
            not_owner: not is_owner
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- raw read and write
    last_blocked: BOOLEAN
        -- True if last read or write call would be blocked
    last_read: INTEGER
        -- how many bytes were read by last call to read
        -- -1 implies last_blocked
    last_written: INTEGER
        -- how many bytes were written by last call to write
        -- -1 implies last_blocked
    read_loop_disabled: BOOLEAN
        -- for certain applications rereading isnt very smart,
        -- i.e. it will block the application. In such cases read just what
        -- is available, do not attempt to read more.  Such file
        -- descriptors do not return an EOF, but just block.
        -- A typical example is a character special file.
    read (buf: POINTER; offset, nbytes: INTEGER)
        -- read data into buf at offset for nbytes bytes.
        -- number of bytes actually read are available in last_read
        require
            can_read: is_open;
            valid_buf: buf /= default_pointer;
            valid_size: nbytes >= 0 -- nbytes < SSIZE_MAX (POSIX value?)
        ensure
            read_no_more_than_requested: last_read <= nbytes
    write (buf: POINTER; offset, nbytes: INTEGER)
        -- write given data from buf at offset, for nbytes bytes.
        require
            can_write: is_open;
            valid_buf: buf /= default_pointer;
            valid_size: nbytes >= 0
        ensure
            wrote_no_more_than_requested: last_written <= nbytes
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- safer read/write
    read_buffer (buf: STDC_BUFFER; offset, nbytes: INTEGER)
        -- more safe version of read in case you have a
        -- STDC_BUFFER object
        require
            dont_overflow: buf.capacity >= offset + nbytes
```

```
    write_buffer (buf: STDC_BUFFER; offset, bytes: INTEGER)
        -- more safe version of write in case you have a
        -- STDC_BUFFER object
        require
            dont_write_garbage: buf.is_valid_index(offset + bytes - 1)
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- buffered input/output, line reading instead of block reading
    last_character: CHARACTER
    last_string: STRING
        -- last read string (includes %N), see STRING_HELPER.chop
    put (a: ANY)
        -- write any Eiffel object as string
    read_character
        -- set last_character.
    read_string (a_size: INTEGER)
        -- implements line reading on top of read. Sets last_string
        -- which includes the new line character if any.
        require
            valid_size: a_size >= 0
    write_character (c: CHARACTER)
    write_string (s: STRING)
    puts (s: STRING)
    put_string (s: STRING)
feature(s) from ABSTRACT_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 ABSTRACT_FILE_DESCRIPTOR
    -- queries
    eof: BOOLEAN
        -- True if end-of-file reached.
        -- Currently Im unsure if detection is reliable
    isatty: BOOLEAN
        -- return true if handle associated with character device
    is_attached_to_terminal: BOOLEAN
        -- return true if handle associated with character device
    is_blocking_io: BOOLEAN
        -- True if blocking i/o enabled (default)
    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
    is_owner: BOOLEAN
        -- does this file descriptor own its descriptor?
        -- only when it owns the descriptor it will automatically close it
    status: POSIX_STATUS
    value: INTEGER
        -- return the value of the file descriptor
        require
            valid_file_descriptor: is_open
feature(s) from ABSTRACT_FILE_DESCRIPTOR
    -- accessible state
    path: STRING
feature(s) from POSIX_FILE_DESCRIPTOR
    -- creation
    make_from_file (file: STDC_FILE)
        -- Create file descriptor from given stream
        -- The stream is leading, so this file descriptor will
        -- never close itself, unless it is made an owner.
        require
            closed: is_closed;
            valid_file: file /= Void and then file.is_open
        ensure
            open: is_open;
            not_owner: not is_owner
feature(s) from POSIX_FILE_DESCRIPTOR
    -- close
    close_on_execute
        -- close this descriptor when forking
feature(s) from POSIX_FILE_DESCRIPTOR
    -- synchronisation
    supports_file_synchronization: BOOLEAN
    supports_data_synchronization: BOOLEAN
    synchronize
        -- synchronize the state of a file (includes synchronize_data)
        require
            synchronize_valid: supports_file_synchronization
    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_data_synchronization
    fdatasync
        -- synchronize the data of a file
        require
            synchronize_valid: supports_data_synchronization
feature(s) from POSIX_FILE_DESCRIPTOR
    -- locking
    get_lock (lock_to_test: POSIX_LOCK): POSIX_LOCK
        -- gets lock information, returns True if a lock is set on
        -- the region in a_lock. a_lock is overwritten with that lock
    set_lock_failed: BOOLEAN
        -- 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
    -- non-blocking i/o
    set_blocking_io (enable: BOOLEAN)
        ensure
            block: enable implies is_blocking_io;
            nonblock: not enable implies not is_blocking_io
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
    accessing_real_singleton: security_is_real_singleton;
    valid_internal_file_descriptor: fd >= - 1;
    valid_open: is_open implies fd >= 0;
    valid_close: not is_open implies fd = - 1;
    valid_status: is_closed implies my_status = Void;
end of POSIX_FILE_DESCRIPTOR
```

D.11 POSIX_FILE_SYSTEM

```
class interface POSIX_FILE_SYSTEM
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from STDC_FILE_SYSTEM
    -- path names
    expand_path (a_path: STRING): STDC_PATH
        -- returns a new path
feature(s) from STDC_FILE_SYSTEM
    -- rename files/directories, remove files/directories
    remove_file (a_path: STRING)
        -- calls unlink when a_path is a file, or rmdir when
        -- a_path is a directory.
        -- error when file could not be removed (and it exists)
        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
        -- 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
feature(s) from ABSTRACT_FILE_SYSTEM
    -- directory access
    change_directory (a_directory: STRING)
        -- Changes the current working directory
```

```
feature(s) from ABSTRACT_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 an empty directory, does not fail if directory
        -- does not exist
        require
            valid_directory_name: a_directory /= Void and then not a_directory.is_empty;
            directory_is_empty: true
    rmdir (a_directory: STRING)
        -- Removes an empty directory, does not fail if directory
        -- does not exist
        require
            valid_directory_name: a_directory /= Void and then not a_directory.is_empty;
            directory_is_empty: true
    force_remove_directory (a_directory: STRING)
        -- Removes a directory, even when not empty
        require
            valid_path: a_directory /= Void and then not a_directory.is_empty
feature(s) from ABSTRACT_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;
            existing_file: is_existing(a_path)
        ensure
            status_returned: Result /= Void
feature(s) from ABSTRACT_FILE_SYSTEM
    -- directory browsing
    browse_directory (a_path: STRING): POSIX_DIRECTORY
        -- Gets information about a directory
        require
            valid_path: a_path /= Void and then not a_path.is_empty;
            path_is_directory: security.error_handling.exceptions_enabled and then status(a_path).is_directory
        ensure
```

```
            directory_returned: Result /= Void
    feature(s) from ABSTRACT_FILE_SYSTEM
        -- accessibility of files
        last_access_result: INTEGER
            -- value of last access test
        is_accessible (a_path: STRING; a_mode: INTEGER): BOOLEAN
            -- Tests for file accessibility
        access (a_path: STRING; a_mode: INTEGER): BOOLEAN
            -- Tests for file accessibility
        is_directory (a_path: STRING): BOOLEAN
            -- return True if a_path exists and if it is a directory
        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_writable (a_path: STRING): BOOLEAN
            -- tests if file is writable by this program
            -- uses real user ID and real group ID instead of effective ones
    feature(s) from ABSTRACT_FILE_SYSTEM
        -- various
        is_case_sensitive: BOOLEAN
            -- is file system case sensitive or not?
        path_separator: CHARACTER
            -- What is the path separator?
    feature(s) from ABSTRACT_FILE_SYSTEM
        -- file system properties
        temporary_directory: STRING
            -- the temporary directory
            ensure
                directory_returned: Result /= Void;
                directory_exists: is_directory(Result);
                directory_is_writable: is_modifiable(Result);
                last_char_not_separator: Result.item(Result.count) /= path_separator
    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
        set_writable (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 times
        touch (a_path: STRING)
            -- Sets the modification and access times of a_path to the
            -- current time of day.
            -- File is created if it does not exist.
        utime (a_path: STRING; access_time, modification_time: POSIX_TIME)
            -- Sets file access and modification times
    feature(s) from POSIX_FILE_SYSTEM
        -- further directory access
        link (existing, new: STRING)
            -- Creates a hard link to a file
            require
                different_names: not existing.is_equal(new)
        unlink (a_path: STRING)
            -- Removes a directory entry, should be a file, not a directory.
            -- its not an error if path does not exist, but all other
            -- errors are reported
            require
                valid_path: a_path /= Void and then not a_path.is_empty
    feature(s) from POSIX_FILE_SYSTEM
        -- mkfifo
        create_fifo (a_path: STRING; a_mode: INTEGER)
            -- Creates 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)
            -- Creates a FIFO special file
```

```
        require
            valid_path: a_path /= Void and then not a_path.is_empty
    feature(s) from POSIX_FILE_SYSTEM
        -- shared memory
        unlink_shared_memory_object (name: STRING)
            -- Remove a shared memory object.
        require
            valid_name: name /= Void and then not name.is_empty
    invariant
        accessing_real_singleton: security_is_real_singleton;
    end of POSIX_FILE_SYSTEM
```

D.12 POSIX_FORK_ROOT

```
deferred class interface POSIX_FORK_ROOT
feature(s) from STDC_CHILD_PROCESS
    -- termination info
    is_terminated: BOOLEAN
        -- True if process 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
feature(s) from ABSTRACT_CHILD_PROCESS
    -- actions that parent may execute
    wait_for (suspend: BOOLEAN)
        -- wait for this process to terminate. If suspend then we
        -- wait until the information about this process is available,
        -- else we return immediately.
        -- If suspend is False, check the running property to see
        -- if this child is really terminated.
        require
            pid_refers_to_child: is_pid_valid;
            not_terminated: not is_terminated
feature(s) from STDC_CURRENT_PROCESS
    -- my stdandard input/output/error
    stdin: POSIX_TEXT_FILE
    stdout: POSIX_TEXT_FILE
    stderr: POSIX_TEXT_FILE
feature(s) from ABSTRACT_CURRENT_PROCESS
    -- process properties
    pid: INTEGER
        -- either the current process identifier or the childs
        require
            valid_pid: is_pid_valid
        ensure
            valid_pid: Result > 0
    is_pid_valid: BOOLEAN
        -- current process id is always valid
feature(s) from ABSTRACT_CURRENT_PROCESS
    -- every process also has standard file descriptors which might not be compatible with stdin/stdout/stderr (Wi
    fd_stdin: POSIX_FILE_DESCRIPTOR
    fd_stdout: POSIX_FILE_DESCRIPTOR
    fd_stderr: POSIX_FILE_DESCRIPTOR
feature(s) from STDC_SECURITY_ACCESSOR
    -- the singleton, available to any because its used in preconditions
    security: STDC_SECURITY
```

```
        -- singleton entry point for security
        ensure
            has_security: Result /= Void
feature(s) from STDC_BASE
    -- errno
    errno: STDC_ERRNO
feature(s) from ABSTRACT_PROCESS
    -- signal this process
    terminate
        -- attempt to gracefully terminate this process
        require
            valid_pid: is_pid_valid
feature(s) from POSIX_PROCESS
    -- signal this process
    kill (a_signal_code: INTEGER)
        -- Send signal signal_code to the process
        require
            valid_pid: is_pid_valid;
            valid_signal: a_signal_code >= 0 -- not_terminated: not is_terminated
feature(s) from POSIX_CURRENT_PROCESS
    -- POSIX locale specifics
    set_native_messages
        -- Select native language as the language in which messages
        -- are displayed
feature(s) from POSIX_FORK_ROOT
    -- process properties
    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
    after_fork
        -- chance for code to do something before the main execute
        -- mainly here for POSIX_DAEMON
    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
        is_signalled: 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_signalled
    invariant
        accessing_real_singleton: security_is_real_singleton;
    end of deferred POSIX_FORK_ROOT
```

D.13 POSIX_GROUP

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

D.14 POSIX_LOCK

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

D.15 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*
-- cleanup
dispose
-- Action to be executed just before garbage collection reclaims an
-- object. (The default action is to do nothing at all.) If you want
-- to change the default action, your class is supposed to
-- inherit MEMORY and to redefine this *dispose* feature.
feature(*s*) **from** *POSIX_MEMORY_MAP*
-- unmap
close
-- remove the mapping
feature(*s*) **from** *POSIX_MEMORY_MAP*
-- state
mem: *STDC_BUFFER*
-- window to memory
offset: *INTEGER*
-- offset from file
size: *INTEGER*
-- number of bytes mapping

```
    fd: POSIX_FILE_DESCRIPTOR
invariant
    accessing_real_singleton: security_is_real_singleton;
end of POSIX_MEMORY_MAP
```

D.16 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_write (allow: BOOLEAN)
            -- give write permission
            ensure
                writability: not allow or allow_owner_write
        set_allow_owner_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
    invariant
        accessing_real_singleton: security_is_real_singleton;
    end of deferred POSIX_PERMISSIONS
```

D.17 POSIX_PIPE

```
class interface POSIX_PIPE
creation
    make
feature(s) from POSIX_PIPE
    -- the pipe
    fdin: POSIX_FILE_DESCRIPTOR
    fdout: POSIX_FILE_DESCRIPTOR
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_pipe: fdin /= Void and fdout /= Void;
end of POSIX_PIPE
```

D.18 POSIX_SEMAPHORE

```
class interface POSIX_SEMAPHORE
feature(s) from POSIX_SEMAPHORE
    -- commands
    attempt_acquire
        -- Lock the semaphore only if it is not locked. If it is locked
        -- by some process, this command returns immediately and the
        -- semaphore is not locked
        require
            initialized: is_initialized;
            unlocked: not is_locked
    acquire
        -- lock the semaphore
        require
            initialized: is_initialized;
            unlocked: not is_locked
    release
        -- unlock the semaphore
        require
            locked: is_locked;
            initialized: is_initialized
        ensure
            unlocked: not is_locked
feature(s) from POSIX_SEMAPHORE
    -- queries
    is_initialized: BOOLEAN
        -- True if semaphore is initialized/opened/created
    is_locked: BOOLEAN
        -- True if this process has locked the semaphore
    supports_semaphores: BOOLEAN
        -- True if semaphores are supported
        -- most systems support unnamed semaphores, but still return False here
    value: INTEGER
        -- value of semaphore if not locked.
        -- Value is <= 0 if this semaphore is locked.
invariant
    accessing_real_singleton: security_is_real_singleton;
    sem_value_valid: sem_value /= Void;
end of POSIX_SEMAPHORE
```

D.19 POSIX_SIGNAL

```
class interface POSIX_SIGNAL
creation
    make (a_value: INTEGER)
        require
            valid_signal: a_value >= 1
feature(s) from POSIX_SIGNAL
    -- creation
    make (a_value: INTEGER)
        require
            valid_signal: a_value >= 1
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
            ignorable: is_ignorable
    set_handler (a_handler: STDC_SIGNAL_HANDLER)
        -- install ones own signal handler
    set_mask (a_mask: POSIX_SIGNAL_SET)
feature(s) from POSIX_SIGNAL
    -- signal functions
    raise_in (a_pid: INTEGER)
        -- raise the signal in the given process
feature(s) from POSIX_SIGNAL
    -- signal state
    child_stop: BOOLEAN
        -- generate SIGCHLD when children stop
    handler: POINTER
        -- pointer to function which catches this signal
    is_defaulted: BOOLEAN
        -- signal is handled by its specific default action
    is_ignored: BOOLEAN
        -- signal is ignored
    is_ignorable: BOOLEAN
        -- True if this signal is ignorable, either it is so by
        -- default or it may be set so.
        ensure
            ignore_consistent: Result implies not is_ignored
    mask: POSIX_SIGNAL_SET
```

```
    refresh
        -- get latest state for this signal
invariant
    accessing_real_singleton: security_is_real_singleton;
    accessing_real_singleton: signal_switch_is_real_singleton;
    valid_signal_value: value >= 1;
    has_memory: sigaction /= Void;
end of POSIX_SIGNAL
```

D.20 POSIX_SIGNAL_SET

```
class interface POSIX_SIGNAL_SET
creation
    make_empty
        -- make an initially empty signal set
    make_full
        -- make a set where all signals are enabled
    make_pending
        -- this signal set will be the set of signals that are blocked
        -- and pending
feature(s) from POSIX_SIGNAL_SET
    -- creation, make a set
    make_empty
        -- make an initially empty signal set
    make_full
        -- make a set where all signals are enabled
    make_pending
        -- this signal set will be the set of signals that are blocked
        -- and pending
feature(s) from POSIX_SIGNAL_SET
    -- change a set
    extend (signo: INTEGER)
        -- add signal to set
        ensure
            is_member: has(signo)
feature(s) from POSIX_SIGNAL_SET
    -- change a set
    put (signo: INTEGER)
        -- add signal to set
        ensure
            is_member: has(signo)
    prune (signo: INTEGER)
        -- remove the signal from the set
        ensure
            is_not_member: not has(signo)
    wipe_out
        -- remove all items
feature(s) from POSIX_SIGNAL_SET
    -- commands to do something with set
    add_to_blocked_signals
        -- Add the signals to the set of blocked signals
    remove_from_blocked_signals
        -- Remove the signals from the set of blocked signals
    set_blocked_signals
        -- Set the set of blocked signals to this set
    suspend
```

```
        -- Suspend process, until delivery of a signal whose action
        -- is either to execute a signal-catching function or to
        -- terminate the process
feature(s) from POSIX_SIGNAL_SET
    -- queries
    has (signo: INTEGER): BOOLEAN
        -- is signal signo in the set
invariant
    accessing_real_singleton: security_is_real_singleton;
    have_set: set /= Void;
end of POSIX_SIGNAL_SET
```

D.21 POSIX_STATUS

```
deferred class interface POSIX_STATUS
feature(s) from POSIX_STATUS
    -- stat members
    is_block_special: BOOLEAN
        -- True if block-special file
    ino: INTEGER
    inode: INTEGER
    permissions: POSIX_PERMISSIONS
        -- file permissions
        ensure
            valid_result: Result /= Void
feature(s) from POSIX_STATUS
    -- direct access to the unix fields, not recommended
    unix_gid: INTEGER
    unix_uid: INTEGER
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_stat: stat /= Void;
end of deferred POSIX_STATUS
```

D.22 POSIX_SYSTEM

class *interface POSIX_SYSTEM*
feature(*s*) **from** *POSIX_SYSTEM*
-- sysconf queries, run-time determined
arg_max: *INTEGER*
-- The lenght 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
invariant
    accessing_real_singleton: security_is_real_singleton;
end of POSIX_SYSTEM
```

D.23 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
    accessing_real_singleton: security_is_real_singleton;
    valid_attr: attr /= Void and then attr.capacity = posix_termios_size;
    valid_fd: fd /= Void;
end of POSIX_TERMIOS
```

D.24 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
    remaining_seconds: INTEGER
        -- number of seconds left in previous request
    seconds: INTEGER
        -- the number of seconds available to execute the command
    set_seconds (a_seconds: INTEGER)
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_seconds: seconds >= 1;
end of deferred POSIX_TIMED_COMMAND
```

D.25 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
    accessing_real_singleton: security_is_real_singleton;
    valid_passwd: passwd /= default_pointer;
end of POSIX_USER
```

D.26 POSIX_USER_DATABASE

```
class interface POSIX_USER_DATABASE
feature(s) from POSIX_USER_DATABASE
    -- queries
    is_existing_uid (uid: INTEGER): BOOLEAN
        -- Returns True if this uid exists in /etc/passwd
        -- (or through NIS or whatever mechanisms that might be in use)
    is_existing_login (login: STRING): BOOLEAN
        -- Returns True if this login exists in /etc/passwd
        -- (or through NIS or whatever mechanisms that might be in use)
invariant
    accessing_real_singleton: security_is_real_singleton;
end of POSIX_USER_DATABASE
```

In this chapter:

E
Short (flat) listing of Single Unix Specification classes

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

E.1 SUS_CONSTANTS

```
class interface SUS_CONSTANTS
feature(s) from SUS_CONSTANTS
    -- syslog facility codes
    LOG_KERN: INTEGER
        -- kernel messages
    LOG_USER: INTEGER
        -- random user-level messages
    LOG_MAIL: INTEGER
        -- mail system
    LOG_DAEMON: INTEGER
        -- system daemons
    LOG_AUTH: INTEGER
        -- security/authorization messages
    LOG_LPR: INTEGER
        -- line printer subsystem
    LOG_NEWS: INTEGER
        -- network news subsystem
    LOG_UUCP: INTEGER
        -- UUCP subsystem
    LOG_CRON: INTEGER
        -- clock daemon
    LOG_LOCAL0: INTEGER
        -- Reserved for local use
    LOG_LOCAL1: INTEGER
        -- Reserved for local use
    LOG_LOCAL2: INTEGER
        -- Reserved for local use
    LOG_LOCAL3: INTEGER
```

```
        -- Reserved for local use
    LOG_LOCAL4: INTEGER
        -- Reserved for local use
    LOG_LOCAL5: INTEGER
        -- Reserved for local use
    LOG_LOCAL6: INTEGER
        -- Reserved for local use
    LOG_LOCAL7: INTEGER
        -- Reserved for local use
feature(s) from SUS_CONSTANTS
    -- syslog open options
    LOG_PID: INTEGER
        -- log the pid with each message
    LOG_CONS: INTEGER
        -- log on the console if errors in sending
    LOG_ODELAY: INTEGER
        -- delay open until first syslog() (default)
    LOG_NDELAY: INTEGER
        -- dont delay open
end of SUS_CONSTANTS
```

E.2 SUS_ENV_VAR

```
class interface SUS_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 SUS_ENV_VAR
    -- commands
    set_value (new_value: STRING)
invariant
    accessing_real_singleton: security_is_real_singleton;
end of SUS_ENV_VAR
```

E.3 SUS_FILE_SYSTEM

```
class interface SUS_FILE_SYSTEM
feature(s) from SUS_FILE_SYSTEM
    -- file statistics
    status (a_path: STRING): SUS_STATUS
        -- Return information about path
        require
            valid_path: a_path /= Void and then not a_path.is_empty;
            existing_file: is_existing(a_path)
        ensure
            status_returned: Result /= Void
    symbolic_link_status (a_path: STRING): SUS_STATUS
        -- Return information about path, but if it is a symbolic
        -- link, about the symbolic link instead of the referenced path
feature(s) from SUS_FILE_SYSTEM
    -- symbolic links
    create_symbolic_link (old_path, new_path: STRING)
        -- Creates a symbolic link
        require
            valid_old: old_path /= Void and then not old_path.is_empty;
            valid_new: new_path /= Void and then not new_path.is_empty
        ensure
            symbolic_link_created: symbolic_link_status(new_path).is_symbolic_link
feature(s) from SUS_FILE_SYSTEM
    -- symbolic links
    symlink (old_path, new_path: STRING)
        -- Creates a symbolic link
        require
            valid_old: old_path /= Void and then not old_path.is_empty;
            valid_new: new_path /= Void and then not new_path.is_empty
        ensure
            symbolic_link_created: symbolic_link_status(new_path).is_symbolic_link
feature(s) from SUS_FILE_SYSTEM
    -- path names
    resolved_path_name (a_path: STRING): STRING
        -- derives from a_path an absolute pathname that names the
        -- same file, whose resolution does not involve ".", "..", or
        -- symbolic links.
feature(s) from SUS_FILE_SYSTEM
    -- path names
    realpath (a_path: STRING): STRING
        -- derives from a_path an absolute pathname that names the
        -- same file, whose resolution does not involve ".", "..", or
        -- symbolic links.
invariant
```

accessing_real_singleton: *security_is_real_singleton*;
end *of SUS_FILE_SYSTEM*

E.4 SUS_HOST

```
class interface SUS_HOST
creation
    make_from_name (a_name: STRING)
        require
            valid_name: a_name /= Void and then not a_name.is_empty
feature(s) from SUS_HOST
    -- creation
    make_from_name (a_name: STRING)
        require
            valid_name: a_name /= Void and then not a_name.is_empty
feature(s) from SUS_HOST
    name: STRING
        -- official (canonical) name of host
    aliases: ARRAY[STRING]
        -- alias names
    address_type: INTEGER
        -- host address type: AF_INET or AF_INET6
    length: INTEGER
        -- length of address: 4 or 16
    addresses: ARRAY[SUS_IP_ADDRESS]
        -- array with IPv4 or IPv6 addresses
invariant
    accessing_real_singleton: security_is_real_singleton;
    has_name: name /= Void and then not name.is_empty;
    has_at_least_one_ip_address: addresses /= Void and addresses.count > 0;
end of SUS_HOST
```

E.5 SUS_SERVICE

```
class interface SUS_SERVICE
creation
    make_from_name (a_name, a_protocol: STRING)
        require
            valid_name: a_name /= Void and then not a_name.is_empty
    make_from_port (a_port: INTEGER; a_protocol: STRING)
        require
            valid_port: a_port >= 0 -- make sure you have a a_protocol if necessary!
feature(s) from SUS_SERVICE
    -- creation
    make_from_name (a_name, a_protocol: STRING)
        require
            valid_name: a_name /= Void and then not a_name.is_empty
    make_from_port (a_port: INTEGER; a_protocol: STRING)
        require
            valid_port: a_port >= 0 -- make sure you have a a_protocol if necessary!
feature(s) from SUS_SERVICE
    -- public state, all data in host byte order
    port: INTEGER
        -- port number
    name: STRING
        -- official service name
    aliases: ARRAY[STRING]
        -- alias list
    protocol: STRING
        -- protocol to use (udp/tcp)
    protocol_type: INTEGER
        -- SOCK_STREAM or SOCK_DGRAM
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_port: port >= 0;
    valid_protocol: protocol /= Void and then protocol.is_equal("tcp") or protocol.is_equal("udp");
    valid_protocol_type: protocol_type = SOCK_STREAM or else protocol_type = SOCK_DGRAM;
end of SUS_SERVICE
```

E.6 SUS_SOCKET_ADDRESS

```
class interface SUS_SOCKET_ADDRESS
creation
    make (a_host: SUS_HOST; a_service: SUS_SERVICE)
        require
            valid_host: a_host /= Void;
            valid_service: a_service /= Void
        ensure
            host_set: host = a_host;
            service_set: service = a_service
feature(s) from SUS_SOCKET_ADDRESS
    -- creation
    make (a_host: SUS_HOST; a_service: SUS_SERVICE)
        require
            valid_host: a_host /= Void;
            valid_service: a_service /= Void
        ensure
            host_set: host = a_host;
            service_set: service = a_service
feature(s) from SUS_SOCKET_ADDRESS
    -- Eiffel features
    host: SUS_HOST
    service: SUS_SERVICE
feature(s) from SUS_SOCKET_ADDRESS
    -- fill socket structure, so ptr returns something valid
    set_address (item: INTEGER)
        -- use one of the ip addresses of host as the socket address
        require
            existing_ip_address_item: item >= host.addresses.lower and item <= host.addresses.upper
feature(s) from SUS_SOCKET_ADDRESS
    -- features the C API calls like
    length: INTEGER
        -- size of my struct sockaddr_in
        ensure
            valid_length: length > 0 -- >= 16 usually
    ptr: POINTER
        -- points to struct sockaddr_in or sockaddr_in6
        ensure
            valid_result: Result /= default_pointer
invariant
    accessing_real_singleton: security_is_real_singleton;
    has_host: host /= Void;
    has_service: service /= Void;
    valid_buf: buf /= Void and then buf.capacity >= length;
end of SUS_SOCKET_ADDRESS
```

E.7 SUS_SYSLOG

```
class interface SUS_SYSLOG
feature(s) from SUS_SYSLOG
    -- open and close
    open (a_identification: STRING; a_format, a_facility: INTEGER)
        -- start logging with the given identification
        require
            valid_identification: a_identification /= Void and then not a_identification.is_empty;
            closed: not is_open
        ensure
            opened: is_open
    close
        -- stop logging
        require
            opened: is_open
        ensure
            closed: not is_open
feature(s) from SUS_SYSLOG
    -- write log messages
    emergency (msg: STRING)
        -- the system is unusable
    alert (msg: STRING)
        -- action must be taken immediately
    critical (msg: STRING)
        -- critical conditions
    error (msg: STRING)
        -- error conditions
    warning (msg: STRING)
        -- warning conditions
    notice (msg: STRING)
        -- normal but significant condition
    info (msg: STRING)
        -- informational
    debug_dump (msg: STRING)
        -- debug-level messages
feature(s) from SUS_SYSLOG
    -- state
    identification: STRING
    format: INTEGER
    facility: INTEGER
    is_open: BOOLEAN
invariant
    accessing_real_singleton: security_is_real_singleton;
    remain_single: Current = the_singleton;
end of SUS_SYSLOG
```

E.8 SUS_TCP_SOCKET

```
class interface SUS_TCP_SOCKET
creation
    open_read_write (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
    open_by_address (sa: SUS_SOCKET_ADDRESS)
feature(s) from SUS_TCP_SOCKET
    -- additional open functions
    open_by_address (sa: SUS_SOCKET_ADDRESS)
invariant
    accessing_real_singleton: security_is_real_singleton;
    valid_internal_file_descriptor: fd >= - 1;
    valid_open: is_open implies fd >= 0;
    valid_close: not is_open implies fd = - 1;
    valid_status: is_closed implies my_status = Void;
end of SUS_TCP_SOCKET
```

F
Short (flat) listing of Standard C bonus classes

In this chapter:

Classes in this appendix are based on Standard C only.

F.1 XML_GENERATOR

```
class interface XML_GENERATOR
creation
    make
    make_with_capacity (a_capacity: INTEGER)
feature(s) from XML_GENERATOR
    -- creation
    make
    make_with_capacity (a_capacity: INTEGER)
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_a_parent (tag: STRING): BOOLEAN
        -- True if tag is a container (parent) at the current level
        -- That can be a just started tag, or a tag higher up
    is_header_written: BOOLEAN
    is_started (tag: STRING): BOOLEAN
        -- True if this tag has just been started
    is_tag_started: BOOLEAN
    is_valid_attribute_name (attribute: STRING): BOOLEAN
        -- Return True if this is a valid attribute name
    unfinished_xml: STRING
        -- the xml in progress
    xml: STRING
        -- the result
```

```
        require
            building_finished: not is_tag_started
feature(s) from XML_GENERATOR
    -- influence state
    clear
        -- start fresh
        ensure
            no_tags: tags.is_empty
feature(s) from XML_GENERATOR
    -- commands that expand xml
    add_header (encoding: STRING)
        require
            valid_point_for_header: not is_header_written
    add_header_iso_8859_1_encoding
    add_header_utf_8_encoding
    add_data (data: STRING)
        -- write data in the current tag
        -- invalid characters like < or > are quoted
        require
            valid_point_for_data: is_tag_started
    add_entity (name: STRING)
    add_raw (raw_data: STRING)
        -- write data straight in the current tag, meta characters
        -- are not quoted.
        require
            valid_point_for_data: is_tag_started
    add_system_doctype (root_tag, system_id: STRING)
        require
            have_header: is_header_written;
            valid_root_tag: root_tag /= Void and then not root_tag.is_empty;
            valid_system_id: system_id /= Void and then not system_id.is_empty
    add_tag (tag, data: STRING)
        -- shortcut for add_tag, add_data and stop_tag
        require
            have_header: is_header_written;
            valid_tag: tag /= Void and then not tag.is_empty
    extend (stuff: STRING)
        -- add anything to the current xml string, youre on your own here!
        -- all changes to my_xml are made from here
    get_attribute (attribute: STRING): STRING
        -- Get contents of attribute attribute for current tag.
        -- Returns Void if attribute doesnt exist
    put_new_line
        -- write a new line at the current position
    put (a: ANY)
        -- write data within the current tag
    puts (stuff: STRING)
```

```
        -- write data within the current tag
    set_attribute (attribute, value: STRING)
        -- Set an attribute of the current tag.
        -- value may not contain an entity reference.
        require
            valid_attribute: is_valid_attribute_name(attribute)
    start_tag (tag: STRING)
        -- start a new tag
        require
            have_header: is_header_written;
            valid_tag: tag /= Void and then not tag.is_empty
    stop_tag
        -- stop last started tag
        require
            tag_is_started: is_tag_started
feature(s) from XML_GENERATOR
    -- quote unsafe characters
    replace_content_meta_characters (s: STRING)
        -- replace all characters in s that have a special meaning in
        -- XML. These characters are < and &
feature(s) from XML_GENERATOR
    -- comments
    start_comment
    stop_comment
invariant
    accessing_real_singleton: security_is_real_singleton;
    same_size: attributes.count = values.count;
    has_tag_stack: tags /= Void;
end of XML_GENERATOR
```

F.2 XHTML_GENERATOR

```
class interface XHTML_GENERATOR
feature(s) from XHTML_GENERATOR
    -- overrule some xml stuff
    new_line_after_closing_tag (a_tag: STRING)
        -- outputs a new line, called when a_tag is closed
        -- can be overridden to start a new line only occasionally
        -- For XHTML documents a new line is treated as a single
        -- space, so it can influence layout.
        require
            valid_tag: a_tag /= Void and then not a_tag.is_empty
    new_line_before_starting_tag (a_tag: STRING)
        -- outputs a new line, called when a_tag is about to begin.
        require
            valid_tag: a_tag /= Void and then not a_tag.is_empty
feature(s) from XHTML_GENERATOR
    -- doctype
    doctype
    doctype_frameset
        -- Output will be frame-based
    doctype_strict
        -- Output will be strict XHTML
    doctype_transitional
        -- Output will be transitional XHTML
feature(s) from XHTML_GENERATOR
    -- page
    b_html
        -- start html page
    e_html
        require
            valid_stop: is_started("html")
feature(s) from XHTML_GENERATOR
    -- header
    meta_refresh_other (time: INTEGER; url: STRING)
    b_head
    e_head
        require
            valid_stop: is_started("head")
    title (a_text: STRING)
        require
            head_started: is_started("head")
feature(s) from XHTML_GENERATOR
    -- body
    b_body
    e_body
        require
```

```
            valid_stop: is_started("body")
feature(s) from XHTML_GENERATOR
    --section headers
    h1 (header_text: STRING)
feature(s) from XHTML_GENERATOR
    -- paragraph
    br
        -- break, the Appendix C.3 way
    b_p
    e_p
        require
            valid_stop: is_started("p")
    p (par: STRING)
feature(s) from XHTML_GENERATOR
    -- layout
    b_tt
        -- teletype writer font
    e_tt
        require
            valid_stop: is_started("tt")
feature(s) from XHTML_GENERATOR
    -- link
    b_a (href: STRING)
        require
            valid_containment: not is_a_parent("a")
    e_a
        require
            valid_stop: is_started("a")
    a (href, s: STRING)
feature(s) from XHTML_GENERATOR
    -- rules
    hr
        -- horizontal rule
feature(s) from XHTML_GENERATOR
    -- white space
    nbsp
        -- non breaking white space
feature(s) from XHTML_GENERATOR
    -- verbatim
    b_pre
    e_pre
        require
            valid_stop: is_started("pre")
feature(s) from XHTML_GENERATOR
    -- tables
    b_table
    e_table
```

```
        require
            valid_stop: is_started("table")
    b_tr
        require
            table_started: is_started("table")
    e_tr
        require
            valid_stop: is_started("tr")
    td
        -- an empty cell
    b_td
        require
            row_started: is_started("tr")
    e_td
        require
            valid_stop: is_started("td")
    th
        -- an empty cell
    b_th
        require
            row_started: is_started("tr")
    e_th
        require
            valid_stop: is_started("th")
feature(s) from XHTML_GENERATOR
    -- forms
    standard_encoding: STRING
    plaintext_encoding: STRING
    multipart_encoding: STRING
    b_form (method, action: STRING)
        require
            valid_method: method /= Void and then not method.is_empty;
            valid_action: action /= Void and then not action.is_empty
    b_form_get (action: STRING)
    b_form_post (action: STRING)
    e_form
        require
            valid_stop: is_started("form")
    b_input (type, name: STRING)
        require
            form_started: is_a_parent("form");
            type_not_empty: type /= Void and then not type.is_empty;
            valid_type: type.is_equal("button") or else type.is_equal("checkbox") or else type.is_equal("file") or else
            name_not_empty: name /= Void and then not name.is_empty
    e_input
        require
            valid_stop: is_started("input")
```

```
    button_hidden (name, value: STRING)
    b_button_submit (name, value: STRING)
    e_button_submit
    b_button_reset
    e_button_reset
    button_reset
    b_checkbox (name, value: STRING)
        require
            have_value: value /= Void and not value.is_empty
    e_checkbox
    b_radio (name, value: STRING)
        require
            have_value: value /= Void and not value.is_empty
    e_radio
    b_select (name: STRING)
    e_select
        require
            valid_stop: is_started("select")
    option (text: STRING)
    selected_option (choice: STRING)
    b_textarea (name: STRING)
        -- multiline input
    e_textarea
        require
            valid_stop: is_started("textarea")
    input_text (name: STRING; size: INTEGER; value: STRING)
        -- singleline input
feature(s) from XHTML_GENERATOR
    -- CSS style sheet support
    b_style
    e_style
        require
            valid_stop: is_started("style")
    set_class (name: STRING)
        -- set attribute class
        require
            valid_name: name /= Void and then not name.is_empty
invariant
    accessing_real_singleton: security_is_real_singleton;
    same_size: attributes.count = values.count;
    has_tag_stack: tags /= Void;
end of XHTML_GENERATOR
```

F.3 EPX_CGI

deferred class *interface EPX_CGI*
feature(*s*) **from** *EPX_CGI*
-- creation
make
feature(*s*) **from** *EPX_CGI*
-- the output routine
execute
-- to be implemented by child
feature(*s*) **from** *EPX_CGI*
-- when script fails
handle_fatal_error
-- your last change to emit something when the script has
-- failed unexpectedly
feature(*s*) **from** *EPX_CGI*
-- debug support
dump_input
-- write cgi input to /tmp/cgi_input
feature(*s*) **from** *EPX_CGI*
-- overrule some xml stuff
extend (*stuff*: *STRING*)
-- output cgi code, never write to *stdout* directly!
feature(*s*) **from** *EPX_CGI*
-- standard variables
auth_type: *STRING*
-- type of authentication used
content_type: *STRING*
-- MIME type of data when invoked with POST method
content_length: *INTEGER*
-- length, in bytes, of data when invoked with POST method
gateway_interface: *STRING*
-- name and version of the gateway, for example CGI/1.1
http_accept: *STRING*
-- contents of the Accept header line sent by the client
http_referer: *STRING*
-- contents of the Referer header line
http_user_agent: *STRING*
-- name of the client program that is making the request
path_info: *STRING*
-- extra path information as it was passed to the server in
-- the query URL
path_translated: *STRING*
-- extra path information translated to a final, usable
-- form. The Web document root is prepended to the query
-- path, and any other path translations are executed.
query_string: *STRING*

```
        -- the input when invoked with the GET method
    remote_addr: STRING
        -- IP address of the client that made the request
    remote_address: STRING
        -- IP address of the client that made the request
    remote_host: STRING
        -- name of the remote computer that made the request
    remote_ident: STRING
        -- user name as given by the ident protocol
    remote_user: STRING
        -- name of the remote user that made the request
    request_method: STRING
        -- name of the method used to invoke the CGI
        -- application. Valid values are GET and POST
    script_name: STRING
        -- name of script that was invoked
    server_name: STRING
        -- domain name of the computer that is running the server software
    server_port: INTEGER
        -- TCP port number on which the server that invoked the CGI
        -- application is operating
    server_protocol: STRING
        -- name of the protocol that the server is using and the
        -- version of that protocol. The protocol name and version
        -- are separated by a forward slash with no spaces, for
        -- instance HTTP/1.0
    server_software: STRING
        -- name of the server that is handling the request
feature(s) from EPX_CGI
    -- standard cgi header
    content_text_html
    content_text_plain
feature(s) from EPX_CGI
    -- server push, multipart header
    content_multipart_x_mixed_replace (boundary: STRING)
        require
            valid_boundary: boundary /= Void and then not boundary.is_empty
    content_next_part
        -- write boundary so next part of multipart msg can be written
        require
            multipart: is_multipart_message
    content_multipart_end
        -- write boundary of multipart
        require
            multipart: is_multipart_message
    is_multipart_message: BOOLEAN
feature(s) from EPX_CGI
```

```
    -- form input
    has_input: BOOLEAN
        -- True if input passed to cgi program
    exist_value (key: STRING): BOOLEAN
        -- returns True if key defined in input
    is_meta_char (c: CHARACTER): BOOLEAN
        -- Return True if c is a commonly used meta characters
    meta_chars: STRING
    raw_value (key: STRING): STRING
        -- returns value for key
        -- if key does not exist, the empty string is returned
    remove_meta_chars (s: STRING)
        -- if s contains meta characters, theyre removed
    value (key: STRING): STRING
        -- returns safe value for key, meta characters are removed
feature(s) from EPX_CGI
    -- Microsoft Internet Explorer stuff
    remove_directory_part (filename: STRING): STRING
        -- IE sends full path instead of just filename as everybody
        -- else does
feature(s) from EPX_CGI
    -- decoding of encoded passed values, thanks Dustin Sallings
    decode_string (in: STRING): STRING
        -- Decode an HTTP encoded string.
    from_hex (in: STRING): CHARACTER
        -- Take a two digit hex string, and convert it to a hex character.
        require
            has_two_digits: in.count = 2 -- in.item(1).is_hexadecimal_digit;
 -- in.item(2).is_hexadecimal_digit;
invariant
    accessing_real_singleton: security_is_real_singleton;
    same_size: attributes.count = values.count;
    has_tag_stack: tags /= Void;
end of deferred EPX_CGI
```

To do

EPX_FILE_SYSTEM

1. Make *EPX_DIRECTORY*.

STDC_FILE

1. read_integer, read_double, read_boolean should perhaps be different for the binary or text files. Now they're satisfy the mico/e definition, so useful for text files only.

STDC_LOCALE_NUMERIC

1. Complete the list of properties

STDC_PATH

1. make some escape char functionality with '%' or so.

POSIX_STATUS

1. return STDC_TIME instead of unix time
2. Not all stat member fields are currently available.

STDC_TIME

1. Add elapsed seconds

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. complete support for nonblocking i/o.

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_SIGNAL

1. Add synchronous waiting for signals like `sigwait`.
2. (Re)enable sending Eiffel exception on signal? i.e. set_exception_handler or so.
3. Resend signal as Eiffel exception in signal handler.

MQUEUE

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

Security

Add base security class that specifies programs intent. Default is to allow anything, but security can be tightened:

1. Call to `open`or `creat`(used?), use real user id, not effective user id.
2. Assume we're free from buffer attacks if preconditions are enabled.
3. exec/system call only allowed when effective user is not root, unless otherwise specified. Or exec only allowed for specific files.
4. Protect against writing specific files/directories. Perhaps substitute vulnerable filenames for other ones.
5. Emulate atomic calls. Or add atomic `access`and `open`call. Shouldn't be done by setting su??
6. When appending/writing to files, check if symbolic link.
7. *ABSTRACT_FILE_SYSTEM.force_remove_directory* is potentially unsafe because it follows links so it can be used to destroy things not under that directory.

8. remove tmpnam function.
9. Make sure the once functions in STDC_BASE are called from within the security initialization, so they're allocated and do not generate an out-of-memory exception themselves.

Idea from 'Remediation of Application Specific Security Vulnerabilities at Runtime' article in IEEE Computer sep/oct 2000.

Windows code

1. chmod also available on Windows.
2. Add permissions to status: read/write.
3. set_binary_mode should do something for the posix factory, i.e., when compiling with cygwin. Perhaps separate *CYGWIN_API* or so in POSIX dir with the window specific stuff.
   Currently cygwin uses text mode for file descriptors, the windows variant uses binary.
4. utime can be supported by using SetFileTime.

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: They have an is_caught function, useful? Means this signal generates an exception.

Known bugs

- not for every *STDC_BASE.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

1 (1996). *System Application Program Interface (API) [C Language]*, volume Part I of *Information technology – Portable Operating System Interface (POSIX)*. ANSI/IEEE, 1996 edition.

2 (1991). *The Standard C library*. Prentice Hall.

3 (1994). *POSIX programmer's guide*. O'Reilly & Associates.

4 (1998). *Unix network programming*. Prentice Hall.

5 (1997). *Object-Oriented Software Construction*. Addison Wesley, 2nd edition.

6 Hatton, L. (2001). Exploring the role of diagnosis in software failure. *IEEE Software*.

7 Whittaker, J. A. (2001). Software's invisible users. *IEEE Software*.

Index