systemp

Portable operating system access protocol.

author:
`Portable Operating-System Interface (POSI) initiative`
version:
`1.1`
date:
`2006/1/21`

compilation:
`static`

(no dependencies on other files)

Public interface

make_directory/1

Makes a new directory. Argument is first expanded to a canonical file name.

compilation:
`static`

template:
`make_directory(Directory)`

mode - number of solutions:
`make_directory(+atom) - one`

exceptions:
Directory is not instantiated:
`instantiation_error`
Directory is neither a variable nor a valid file name:
`type_error(file_name, Directory)`
No permission for making a new directory:
`permission_error(write, Directory)`

delete_directory/1

Deletes an empty directory.

compilation:
`static`

template:
`delete_directory(Directory)`

mode - number of solutions:
`delete_directory(+atom) - one`

exceptions:
Directory is not instantiated:
`instantiation_error`
Directory is neither a variable nor a valid file name:
`type_error(file_name, Directory)`
Directory does not exists:
`existence_error(directory, Directory)`
No permission for deleting the directory:
`permission_error(write, Directory)`
Directory is not empty:
`permission_error(write, Directory)`

change_directory/1

Changes current working directory.

compilation:
`static`

template:
`change_directory(Directory)`

mode - number of solutions:
`change_directory(+atom) - one`

exceptions:
Directory is not instantiated:
`instantiation_error`
Directory is neither a variable nor a valid file name:
`type_error(file_name, Directory)`
No permission for accessing the directory:
`permission_error(read, Directory)`
Directory does not exists:
`existence_error(directory, Directory)`

working_directory/1

Current working directory (as an absolute file name).

compilation:
`static`

template:
`working_directory(Directory)`

mode - number of solutions:
`working_directory(?atom) - zero_or_one`

exceptions:
Directory is neither a variable nor a valid file name:
`type_error(file_name, Directory)`

directory_exists/1

True if the specified directory exists (irrespective of directory permissions).

compilation:
`static`

template:
`directory_exists(Directory)`

mode - number of solutions:
`directory_exists(+atom) - zero_or_one`

exceptions:
Directory is not instantiated:
`instantiation_error`
Directory is neither a variable nor a valid file name:
`type_error(file_name, Directory)`

directory_files/3

List of all directory files that matches a regular expression (returns an empty list when no file matches; may be used to find hidden files given an appropriate filter).

compilation:
`static`

template:
```
directory_files(Directory, Filter, Files)
```
mode - number of solutions:
```
directory_files(+atom, +atom, -list) - one
```
exceptions:
- Directory is not instantiated:
  ```
  instantiation_error
  ```
- Directory is neither a variable nor a valid file name:
  ```
  type_error(file_name, Directory)
  ```
- No read permission for the directory:
  ```
  permission_error(read, Directory)
  ```
- Directory does not exists:
  ```
  existence_error(directory, Directory)
  ```
- Filter is not instantiated:
  ```
  instantiation_error
  ```
- Filter is neither a variable nor a valid regular expression:
  ```
  type_error(regular_expression, Filter)
  ```

directory_files/2

List of all directory files (returns an empty list when the directory is empty; hidden files are not retrieved).

compilation:
```
static
```
template:
```
directory_files(Directory, Files)
```
mode - number of solutions:
```
directory_files(+atom, -list) - one
```
exceptions:
- Directory is not instantiated:
  ```
  instantiation_error
  ```
- Directory is neither a variable nor a valid file name:
  ```
  type_error(file_name, Directory)
  ```
- No read permission for the directory:
  ```
  permission_error(read, Directory)
  ```
- Directory does not exists:
  ```
  existence_error(directory, Directory)
  ```

delete_file/1

Deletes a file.

compilation:
```
static
```
template:
```
delete_file(File)
```
mode - number of solutions:
```
delete_file(+atom) - one
```
exceptions:
- File is not instantiated:
  ```
  instantiation_error
  ```
- File is neither a variable nor a valid file name:
  ```
  type_error(file_name, File)
  ```
- File does not exists:
  ```
  existence_error(file, File)
  ```

No write permission to the file:
```
permission_error(write, File)
```

delete_files/1

Deletes a set of matching files.

compilation:
```
static
```

template:
```
delete_files(Filter)
```

mode - number of solutions:
```
delete_files(+atom) - one
```

exceptions:
Filter is not instantiated:
```
instantiation_error
```
Filter is neither a variable nor a valid regular expression:
```
type_error(regular_expression, Filter)
```
No permission to delete some of the matching files:
```
permission_error(write, File)
```

rename_file/2

Renames a file (or a directory).

compilation:
```
static
```

template:
```
rename_file(Old, New)
```

mode - number of solutions:
```
rename_file(+atom, +atom) - zero_or_one
```

exceptions:
Old is not instantiated:
```
instantiation_error
```
New is not instantiated:
```
instantiation_error
```
Old is neither a variable nor a valid file name:
```
type_error(file_name, Old)
```
New is neither a variable nor a valid file name:
```
type_error(file_name, New)
```
File Old does not exists:
```
existence_error(file, Old)
```
No write permission to the file:
```
permission_error(write, Old)
```

copy_file/2

Makes a copy of a file.

compilation:
```
static
```

template:
```
copy_file(Original, Copy)
```

mode - number of solutions:
```
copy_file(+atom, +atom) - one
```

exceptions:
Original is not instantiated:
```
instantiation_error
```
Copy is not instantiated:
```
instantiation_error
```
Original is neither a variable nor a valid file name:
```
type_error(file_name, Original)
```
Copy is neither a variable nor a valid file name:
```
type_error(file_name, Copy)
```
File Original does not exists:
```
existence_error(file, Original)
```
No read permission to the original file:
```
permission_error(read, Original)
```
No write permission to the file copy:
```
permission_error(write, Copy)
```

make_symlink/2

Makes a symbolic link.

compilation:
```
static
```
template:
```
make_symlink(Symlink, Target)
```
mode - number of solutions:
```
make_symlink(+atom, +atom) - one
```
exceptions:
Symlink is not instantiated:
```
instantiation_error
```
Target is not instantiated:
```
instantiation_error
```
Symlink is neither a variable nor a valid file name:
```
type_error(file_name, Symlink)
```
Target is neither a variable nor a valid file name:
```
type_error(file_name, Target)
```
No permission for creating the symbolic link:
```
permission_error(write, Symlink)
```

file_exists/1

True if the specified file exists (irrespective of type and file permissions).

compilation:
```
static
```
template:
```
file_exists(File)
```
mode - number of solutions:
```
file_exists(+atom) - zero_or_one
```
exceptions:
File is not instantiated:
```
instantiation_error
```
File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```

file_property/2

File properties.

compilation:
```
static
```
template:
```
file_property(File, Property)
```
mode - number of solutions:
```
file_property(+atom, +compound) - zero_or_one
file_property(+atom, -compound) - one_or_more
```
exceptions:

File is not instantiated:
```
instantiation_error
```
File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```
File does not exists:
```
existence_error(file, File)
```
No read permission to the file:
```
permission_error(read, File)
```
Property is neither a variable nor a valid file property:
```
type_error(file_property, Property)
```
examples:

Querying file size:
```
file_property(foo, size(A))
A=32568
```
Querying file type:
```
file_property(foo, type(A))
A=regular
```
Querying file creation date:
```
file_property(foo, creation_time(A))
A=137692
```
Querying file last access date:
```
file_property(foo, access_time(A))
A=811042
```
Querying file modification date:
```
file_property(foo, modification_time(A))
A=811042
```
Querying file permissions:
```
file_property(foo, permission(A))
A=read
```
current_environment_variable/1

Argument is a currently defined environment variable . Fails if the variable does not exists.

compilation:
```
static
```
template:
```
current_environment_variable(Variable)
```
mode - number of solutions:
```
current_environment_variable(?atom) - zero_or_more
```
exceptions:

Variable is neither a variable nor an atom:
```
type_error(atom, Variable)
```
delete_environment_variable/1

Deletes an environment variable.

compilation:
`static`

template:
`delete_environment_variable(Variable)`

mode - number of solutions:
`delete_environment_variable(+atom) - one`

exceptions:
Variable is not instantiated:
`instantiation_error`
Variable is neither a variable nor an atom:
`type_error(atom, Variable)`
Variable is not a currently defined environment variable:
`existence_error(environment_variable, Variable)`

get_environment_variable/2

Gets environment variable value.

compilation:
`static`

template:
`get_environment_variable(Variable, Value)`

mode - number of solutions:
`get_environment_variable(+atom, ?atom) - zero_or_one`

exceptions:
Variable is not instantiated:
`instantiation_error`
Variable is neither a variable nor an atom:
`type_error(atom, Variable)`
Value is neither a variable nor an atom:
`type_error(atom, Value)`
Variable is not a currently defined environment variable:
`existence_error(environment_variable, Variable)`

set_environment_variable/2

Sets environment variable value.

compilation:
`static`

template:
`set_environment_variable(Variable, Value)`

mode - number of solutions:
`set_environment_variable(+atom, +atom) - one`

exceptions:
Variable is not instantiated:
`instantiation_error`
Value is not instantiated:
`instantiation_error`
Variable is neither a variable nor an atom:
`type_error(atom, Variable)`
Value is neither a variable nor an atom:
`type_error(atom, Value)`

time_stamp/1

Returns a system-dependent time stamp (which can be used for sorting).

compilation:
```
static
```
template:
```
time_stamp(Time)
```
mode - number of solutions:
```
time_stamp(-number) - one
```

local_time/1

Local time (respecting time zone and daylight savings settings).

compilation:
```
static
```
template:
```
local_time(time(Year, Month, Day, Hours, Mins, Secs, Microsecs))
```
mode - number of solutions:
```
local_time(?time(?integer, ?integer, ?integer, ?integer, ?integer, ?integer,
?integer)) - zero_or_one
```

utc_time/1

Universal Coordinated Time (UTC).

compilation:
```
static
```
template:
```
utc_time(time(Year, Month, Day, Hours, Mins, Secs, Microsecs))
```
mode - number of solutions:
```
utc_time(?time(?integer, ?integer, ?integer, ?integer, ?integer, ?integer,
?integer)) - zero_or_one
```

convert_time/2

Converts between system-dependent time stamps and calendar local date and time.

compilation:
```
static
```
template:
```
convert_time(Time, time(Year, Month, Day, Hours, Mins, Secs, Microsecs))
```
mode - number of solutions:
```
convert_time(+number, ?time(?integer, ?integer, ?integer, ?integer, ?integer,
?integer, ?integer)) - zero_or_one
convert_time(?number, +time(+integer, +integer, +integer, +integer, +integer,
+integer, +integer)) - zero_or_one
```
exceptions:

Neither argument is instantiated:
```
instantiation_error
```
Time stamp is neither a variable nor a valid time stamp:
```
type_error(time_stamp, Time)
```
Time structure is neither a variable nor a valid time structure:
```
type_error(time_structure, time(Year, Month, Day, Hours, Mins, Secs,
Microsecs))
```

cpu_time/1

System cpu time in seconds.

compilation:
```
static
```

template:
```
cpu_time(Time)
```
mode - number of solutions:
```
cpu_time(-number) - one
```

host_name/1

Host name (default is localhost).

compilation:
```
static
```
template:
```
host_name(Name)
```
mode - number of solutions:
```
host_name(-atom) - one
```

portable_os_file_name/2

Converts between portable and operating-system dependent file names.

compilation:
```
static
```
template:
```
portable_os_file_name(Canonical, OS)
```
mode - number of solutions:
```
portable_os_file_name(+atom, -atom) - one
portable_os_file_name(-atom, +atom) - one
```

portable_file_name/3

Converts between relative, absolute, and URL portable file names.

compilation:
```
static
```
template:
```
portable_file_name(Relative, Absolute, URL)
```
mode - number of solutions:
```
portable_file_name(+atom, -atom, -atom) - one
portable_file_name(-atom, +atom, -atom) - one
portable_file_name(-atom, -atom, +atom) - one
```
exceptions:

None of the arguments is instantiated:
```
instantiation_error
```
Relative is neither a variable nor a relative file name:
```
type_error(relative_file_name, Relative)
```
Absolute is neither a variable nor a absolute file name:
```
type_error(absolute_file_name, Absolute)
```
URL is neither a variable nor a file name URL:
```
type_error(url_file_name, URL)
```

relative_file_name/1

True when the argument is a valid, relative file name. Argument is expanded to a canonical file name before testing.

compilation:
```
static
```

template:
```
relative_file_name(File)
```
mode - number of solutions:
```
relative_file_name(+atom) - zero_or_one
```
exceptions:
File is not instantiated:
```
instantiation_error
```
File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```

absolute_file_name/1

True if the argument is a valid, absolute file name. Argument is expanded to a canonical file name before testing.

compilation:
```
static
```
template:
```
absolute_file_name(File)
```
mode - number of solutions:
```
absolute_file_name(+atom) - zero_or_one
```
exceptions:
File is not instantiated:
```
instantiation_error
```
File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```

url_file_name/1

True when the argument is a valid, URL file name. Argument is expanded to a canonical file name before testing.

compilation:
```
static
```
template:
```
url_file_name(File)
```
mode - number of solutions:
```
url_file_name(+atom) - zero_or_one
```
exceptions:
File is not instantiated:
```
instantiation_error
```
File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```

absolute_file_name/2

Expands a file name into a canonical absolute file name.

compilation:
```
static
```
template:
```
absolute_file_name(File, Absolute)
```
mode - number of solutions:
```
absolute_file_name(+atom, ?atom) - zero_or_one
```
exceptions:
File is not instantiated:
```
instantiation_error
```

File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```
Absolute is neither a variable nor a valid file name:
```
type_error(file_name, Absolute)
```

url_file_name/2

Expands a file name into a canonical URL file name.

compilation:
```
static
```

template:
```
url_file_name(File, URL)
```

mode - number of solutions:
```
url_file_name(+atom, ?atom) - zero_or_one
```

exceptions:
File is not instantiated:
```
instantiation_error
```
File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```
URL is neither a variable nor a valid file name URL:
```
type_error(file_name, URL)
```

file_name_part/2

File name parts. The file name is expanded to a canonical file name before decomposing in parts.

compilation:
```
static
```

template:
```
file_name_part(File, Part)
```

mode - number of solutions:
```
file_name_part(+atom, ?compound) - zero_or_more
```

exceptions:
File is not instantiated:
```
instantiation_error
```
File is neither a variable nor a valid file name:
```
type_error(file_name, File)
```
File does not exists:
```
existence_error(file, File)
```
Part is neither a variable nor a file name part:
```
type_error(file_name_part, Port)
```

examples:
Querying file access protocol:
```
file_name_part(foo, protocol(A))
A=file
```

Querying file host location:
```
file_name_part('http://www.prolog-standard.org:8080/index.html', host(A))
A='www.prolog-standard.org'
```

Querying file port:
```
file_name_part('http://www.prolog-standard.org:8080/index.html', port(A))
A=8080
```

Querying file port:
```
file_name_part(foo, port(A))
no
```

Querying file username:

```
file_name_part('http://user@www.prolog-standard.org/', user(A))
A=user
```

Querying file password:

```
file_name_part('http://user:password@www.prolog-standard.org/',
password(A))
A=password
```

Querying file base name:

```
file_name_part('/usr/local/foo.pl', base(A))
A='foo.pl'
```

Querying file path:

```
file_name_part('/usr/local/foo.pl', path(A))
A='/usr/local/'
```

Querying file extension:

```
file_name_part('foo.pl', extension(A))
A='.pl'
```

Querying file extension:

```
file_name_part('foo.', extension(A))
A='.'
```

Querying file extension:

```
file_name_part(foo, extension(A))
A=
```

Querying file search pairs:

```
file_name_part('http://user@www.prolog-standard.org/
updates.cgi?date=today', search(A))
A=[date=today]
```

Querying file fragment:

```
file_name_part('http://user@www.prolog-standard.org/updates.html#latest',
fragment(A))
A=latest
```

file_name_parts/2

Converts between a file name and its constituent parts (represented as a list of compound terms). The file name (when instantiated) is expanded to a canonical file name before decomposing in parts.

compilation:

```
static
```

template:

```
file_name_parts(File, Parts)
```

mode - number of solutions:

```
file_name_parts(+atom, -list(compound)) - one
file_name_parts(-atom, +list(compound)) - zero_or_one
```

exceptions:

None of the arguments are instantiated:

```
instantiation_error
```

File is neither a variable nor a valid file name:

```
type_error(file_name, File)
```

Parts is neither a variable nor a list:

```
type_error(list(compound), Parts)
```

examples:

Decomposing a file name:

```
file_name_parts('http://www.prolog-standard.org:8080/index.html', Parts)
Parts=[protocol(http), host('www.prolog-standard.org'), port(8080),
path('/'), base(index), extension('.html')]
```

Protected interface

(none)

Private predicates

(none)

Remarks

File names overview:
: The main idea is that file names should be operating-system independent. As such, predicates are needed to convert between portable file names and operating-system specific file names. The solution chosen is to use URL syntax for portable file names.

Local and remote file names:
: A (portable) file name may point to either a local file or a remote file.

URL file names:
: These are file names which start with an access protocol (e.g. {http, https, ftp, gopher, file}://).

Absolute file names:
: These are file names that always point to a local file. They always start with a slash character (/).

Relative file names:
: These are file names that always point to a local file. A file name is a relative file name if it does not start with a slash character or a file access protocol (including the :// characters).

Canonical file names
: These are file names where any environment variables was been expanded and where the sequences for current (.) and parent (..) directories have been resolved.

Time stamps:
: Time stamps are used for representing current, system time and in file properties to represent creation, modification, and access times. Time stamps are system-dependent terms but that can be compared (e.g. when testing which of two given files is older).