Module ThreadUnix

```
module ThreadUnix: sig .. end
```

Deprecated. The functionality of this module has been merged back into the `Unix` module. Threaded programs can now call the functions from module `Unix` directly, and still get the correct behavior (block the calling thread, if required, but do not block all threads in the process).
Thread-compatible system calls.

Process handling

```
val execv : string -> string array -> unit
```

```
val execve : string -> string array -> string array -> unit
```

```
val execvp : string -> string array -> unit
```

```
val wait : unit -> int * Unix.process_status
```

```
val waitpid : Unix.wait_flag list -> int -> int * Unix.process_status
```

```
val system : string -> Unix.process_status
```

Basic input/output

```
val read : Unix.file_descr -> string -> int -> int -> int
```

```
val write : Unix.file_descr -> string -> int -> int -> int
```

Input/output with timeout

```
val timed_read : Unix.file_descr -> string -> int -> int -> float -> int
```

See `ThreadUnix.timed_write`.

```
val timed_write : Unix.file_descr -> string -> int -> int -> float -> int
```

Behave as `ThreadUnix.read` and `ThreadUnix.write`, except that `Unix_error(ETIMEDOUT,_,_)` is raised if no data is available for reading or ready for writing after `d` seconds. The delay `d` is given in the fifth argument, in seconds.

Polling

```
val select : Unix.file_descr list ->
       Unix.file_descr list ->
       Unix.file_descr list ->
```

```
float -> Unix.file_descr list * Unix.file_descr list * Unix.file_descr list
```

Pipes and redirections

```
val pipe : unit -> Unix.file_descr * Unix.file_descr
```

```
val open_process_in : string -> in_channel
```

```
val open_process_out : string -> out_channel
```

```
val open_process : string -> in_channel * out_channel
```

Time

```
val sleep : int -> unit
```

Sockets

```
val socket : Unix.socket_domain -> Unix.socket_type -> int -> Unix.file_descr
```

```
val accept : Unix.file_descr -> Unix.file_descr * Unix.sockaddr
```

```
val connect : Unix.file_descr -> Unix.sockaddr -> unit
```

```
val recv : Unix.file_descr -> string -> int -> int -> Unix.msg_flag list -> int
```

```
val recvfrom : Unix.file_descr ->
       string -> int -> int -> Unix.msg_flag list -> int * Unix.sockaddr
```

```
val send : Unix.file_descr -> string -> int -> int -> Unix.msg_flag list -> int
```

```
val sendto : Unix.file_descr ->
       string -> int -> int -> Unix.msg_flag list -> Unix.sockaddr -> int
```

```
val open_connection : Unix.sockaddr -> in_channel * out_channel
```