Turbo LUA

Getting ioloop instance:

```
local turbo = require 'turbo'
local tio = turbo.ioloop.instance()
```

There is only one instance at time. Multiple requests always return the same instance.

Always use pcall

For example: if you register periodic action using

```
tio:set_interval()
```

and registered handler fails from some reason, it will be removed by tio.

ioloop.lua

> Because errors can always happen! If a handler errors, the handler is removed from the IOLoop, and never called again.

tio:add_handler

Add handler function for given event mask on fd

```
function MyClass:procesData()
...
end
function MyClass:init()
  tio:add_handler(TCPSocketFd, turbo.ioloop.READ, self.processData, self)
end
```

Internally function is performing:

```
self._poll:register()
self.handlers[fs] = {handler, arg}
```

When FD is closed:

- Q6 Will the close of an fd cause it to be removed from all epoll sets automatically?
- A6 Yes.

But tio behaves in stupid way, and starts calling handler in unfinished loop. Registered FD has to be manually removed:

```
tio:remove_handler(TCPSocketFd)
```

It will show short notice about already removed socket which can be ignored: `[ioloop.lua] unregister() in remove_handler() failed: Bad file descriptor`

tio:set_interval

```
tio:set_interval (10000, self.method, self)
local ref = tio:set_interval (10000, self.method, self)
tio:clear_interval (ref)
```

tio:add_timeout

```
tio:add_timeout (turbo.util.gettimemonotonic() + 10000, self.method, self)
local ref = tio:add_timeout (turbo.util.gettimemonotonic() + 10000,
self.method, self)
tio:remove_timeout (ref)
```