# proty Documentation

***Release 0.4***

**2010-2012, Thomas Gatzweiler**

January 14, 2012

# CONTENTS

Last updated: January 14, 2012 (v0.4)

Contents:

CHAPTER

ONE

# TUTORIAL

## 1.1 Installation

In order to install proty you should check if there is a package available for your operation system or linux distribution. If not you have to build proty from source (see *From Source*).

### 1.1.1 Arch Linux

There is a build script in the AUR to build the most recent proty release. If you have yaourt installed you can install proty by invoking this command:

```
$ yaourt -S proty
```

If not you can install proty like this:

```
$ mkdir proty && cd proty
$ wget http://aur.archlinux.org/packages/pr/proty/PKGBUILD
$ makepkg -i
```

### 1.1.2 From Source

To build proty from source you have to download the latest source release from proty.cc. Extract it and execute the following commands:

```
$ mkdir build && cd build
$ cmake /path/to/source
$ make
$ sudo make install
```

CHAPTER

TWO

# LANGUAGE REFERENCE

## 2.1 Expressions

## 2.2 Basic Objects

### 2.2.1 String

Strings are written in double quotes:

```
"This is a simple string"
```

#### Escape sequences

Strings can contain the following escape sequences:

| sequence | function |
|---|---|
| `\a` | bell |
| `\b` | backspace |
| `\f` | formfeed |
| `\n` | linefeed |
| `\r` | carriage return |
| `\t` | horizontal tab |
| `\v` | vertical tab |
| `\xhh` | custom with the hex value *hh* |

#### Methods

**`String.+(other)`**
Appends another **String** to the string.

**`String.==(other)`**
Returns `true` if the string equals the **String** `other`.

**`String.!=(other)`**
Returns `true` if the string doesn't equal the **String** `other`.

`String.`**`bool`**`()`
Returns `true` if the length of the string is greater than zero, otherwise `false`.

`String.`**`length`**`()`
: Returns the length of the string as **Integer**.

`String.`**`split`**`(`*sep*`)`
: Splits the string at occurencies of `sep` and returns a **List** of the seperated strings.

## 2.3 Modules

CHAPTER

THREE

# LIBRARY REFERENCE

## 3.1 `io` — Input/Output functions

### 3.1.1 Output

`io.`**`print`**(*obj*)
Prints an `obj` to *stdout*.

### 3.1.2 File handling

`io.`**`open`**(*filename*, *mode*)
Opens a file with the specificied mode. Returns a **File** object.

`File.`**`read`**(*len*)
Returns `len` bytes of the file.

`File.`**`write`**(*str*)
Writes `str` to the file.

`File.`**`close`**()
Closes the file.

## 3.2 `os` — Operating system interfaces

### 3.2.1 Environment variables

`os.`**`getenv`**(*varname*)
Returns the value of the environment variable `varname`.

`os.`**`setenv`**(*varname*, *value*)
Sets the environment variable `varname` to `value`.

### 3.2.2 Process management

`os.`**`system`**(*command*)
Execute `command` in a subshell. Returns the status code of the command.

`os.`**`exit`**(*status*)
Exit with the specified `status` code.

## 3.3 `net` — Network functions

### 3.3.1 Sockets

`net.`**`socket`**`()`
Returns a **Socket** object.

`Socket.`**`connect`**`(`*host*, *port*`)`
Connects to the specified host on the given port.

`Socket.`**`send`**`(`*str*`)`
Sends `str` to the socket.

`Socket.`**`recv`**`(`*len*`)`
Receives data from the socket. The maximum amount of data is specified by `len`.

## 3.4 `math` — Mathematical functions

### 3.4.1 Power and logarithmic functions

`math.`**`log`**`(`*x*`)`
Returns the natural logarithm of *x* (to base *e*).

`math.`**`pow`**`(`*x*, *y*`)`
Returns `x` to the power of `y`.

`math.`**`sqrt`**`(`*x*`)`
Returns the square root of `x`.

### 3.4.2 Trigonometric functions

`math.`**`sin`**`(`*x*`)`
Returns the sine of `x` radians.

`math.`**`cos`**`(`*x*`)`
Returns the cosine of `x` radians.

`math.`**`tan`**`(`*x*`)`
Returns the tangent of `x` radians.

`math.`**`acos`**`(`*x*`)`
Returns the arc cosine of `x`, in radians.

`math.`**`asin`**`(`*x*`)`
Returns the arc sine of `x`, in radians.

`math.`**`atan`**`(`*x*`)`
Returns the arc tangent of `x`, in radians.

### 3.4.3 Constants

## 3.5 `time` — Time functions

`time.`**`sleep`**(*seconds*)
Suspends the program for the given time.

`time.`**`now`**()
Returns a **Time** object.

`Time.`**`strftime`**(*format*)
Returns a string in the specified format.

CHAPTER

FOUR

# C API

## 4.1 Introduction

Proty offers a very simple C API because the runtime library is written entirely in C.

### 4.1.1 Include Files

To get access to all necessary functions and definitions include the following header:

```
#include <proty/runtime.h>
```

### 4.1.2 Create a Module

A module must contain a `foo_init()` function where *foo* is the name of the module:

```
Object* foo_init() {
  Object* foo = Object_new(Object_proto);
  return foo;
}
```

This function will be called with the `load("foo")` function in a proty program:

```
foo = load("foo")
```

`foo` is now the object returned by the `foo_init()` function.

CHAPTER

FIVE

# INDICES AND TABLES

- *genindex*
- *search*