Squirrel Standard Library 1.0

Alberto Demichelis

Squirrel Standard Library 1.0

Alberto Demichelis

Table of Contents

Chapter 1. Introduction

The squirrel standard libraries consist in a set of modules implemented in C++. While are not essential for the language, they provide a set of useful services that are commonly used by a wide range of applications(file I/O, dynamic modules loading, etc...), plus they offer a foundation for developing additional libraries.

All libraries are implemented through the squirrel API and the ANSI C runtime library. The modules are organized in the following way:

- I/O : input and output
- blob : binary buffers manipilation
- math : basic mathematical routines
- system : system access function
- string : string formatting and manipulation
- module : dynamic modules loading

The libraries can be registered independently,except for the IO library that depends from the bloblib.

Chapter 2. The Input/Output library

the input lib implements basic input/output routines.

Squirrel API

Global functions

`fopen(`*`filename`*`, `*`mode`*`)`.
opens or create a file and retuns it as file object*`filename`* is the absolute or realtive path of the file, *`mode`* is the access mode. This string is exactly what is used in the standard C function `fopen`.

File object

`readstr(size, [encoding])`.
reads n characters from the stream and returns it as string. if *`encoding`* is omitted the function read an 8bit string. *`encoding`* can have the following values:

`'a'`	8bits character string
`'u'`	16bits character string

`readblob(size)`.
read n bytes from the stream and retuns them as blob

`readn(type)`.
reads a number from the stream according to the type pameter. *`type`* can have the following values:

`'i'`	32bits number	returns an integer
`'s'`	16bits signed integer	returns an integer
`'w'`	16bits unsigned integer	returns an integer
`'c'`	8bits signed integer	returns an integer
`'b'`	8bits unsigned integer	returns an integer
`'f'`	32bits float	returns an float
`'d'`	64bits float	returns an float

`writestr(`*`str`*`, `*`[encoding]`*`)`.
writes a string in the stream *`str`* is the string that as to be written, *`encoding`* is and optional parameter that can have the following values:

`'a'`	8bits character string
`'u'`	16bits character string

if *`encoding`* is omitted the default value is `'a'`.

`writeblob(blob)`.
writes a blob in the stream

`writen(n, type)`.
writes a number in the stream formatted according to the type pameter. *`type`* can have the following values:

`'i'`	32bits number
`'s'`	16bits signed integer
`'w'`	16bits unsigned integer
`'c'`	8bits signed integer

`'b'`	8bits unsigned integer
`'f'`	32bits float
`'d'`	64bits float

`seek(offset, [origin])`.
Moves the read/write pointer to a specified location. *`offset`* indicates the number of bytes from *`origin`*. *`origin`* can be `'b'` beginning of the stream,`'c'` current location or `'e'` end of the stream. If origin is omitted the parameter is defaulted as 'b'(beginning of the stream).

`tell()`.
returns read/write pointer absolute position

`len()`.
returns the lenght of the stream

`eos()`.
returns a non null value if the read/write pointer is at the end of the stream.

C API

!!FIXME!!

Chapter 3. The Blob library

The blob library implements binary data manipulations routines. The library is based on `blob objects` that rapresent a buffer of arbitrary binary data.

Squirrel API

Global functions

`blob(`*`size`*`)`.
returns a new blob object of the specified size in bytes

`rawcastI2F(`*`n`*`)`.
casts a int to a float

`rawcastF2I(`*`f`*`)`.
casts a float to a int

`swap2(`*`n`*`)`.
swap the byte order of a number (like it would be a 16bits integer)

`swap4(`*`n`*`)`.
swap the byte order of an integer

`swapfloat(`*`f`*`)`.
swaps the byteorder of a float

Blob object

The blob object is a buffer of arbitrary binary data. The object behaves like a file stream, it has a read/write pointer and it automatically grows if data is written out of his boundary.
A blob can also be accessed byte by byte through the `[]` operator.

`readstr(size, [encoding])`.
reads n characters from the stream and returns it as string. if *`encoding`* is omitted the function read an 8bit string. *`encoding`* can have the following values:

`'a'`	8bits character string
`'u'`	16bits character string

`readblob(size)`.
read n bytes from the stream and retuns them as blob

`readn(type)`.
reads a number from the stream according to the type pameter. *`type`* can have the following values:

`'i'`	32bits number	returns an integer
`'s'`	16bits signed integer	returns an integer
`'w'`	16bits unsigned integer	returns an integer
`'c'`	8bits signed integer	returns an integer
`'b'`	8bits unsigned integer	returns an integer
`'f'`	32bits float	returns an float
`'d'`	64bits float	returns an float

`writestr(`*`str`*`, `*`[encoding]`*`)`.
writes a string in the stream *`str`* is the string that as to be written, *`encoding`* is and optional parameter

that can have the following values:

`'a'`	8bits character string
`'u'`	16bits character string

if *`encoding`* is omitted the default value is `'a'`.

`writeblob(blob)`.
writes a blob in the stream

`writen(n, type)`.
writes a number in the stream formatted according to the type pameter. *`type`* can have the following values:

`'i'`	32bits number
`'s'`	16bits signed integer
`'w'`	16bits unsigned integer
`'c'`	8bits signed integer
`'b'`	8bits unsigned integer
`'f'`	32bits float
`'d'`	64bits float

`seek(offset, [origin])`.
Moves the read/write pointer to a specified location. *`offset`* indicates the number of bytes from *`origin`*. *`origin`* can be `'b'` beginning of the stream,`'c'` current location or `'e'` end of the stream. If origin is omitted the parameter is defaulted as 'b'(beginning of the stream).

`tell()`.
returns read/write pointer absolute position

`len()`.
returns the lenght of the stream

`eos()`.
returns a non null value if the read/write pointer is at the end of the stream.

`resize(`*`size`*`)`.
resizes the blob to the specified *`size`*

`swap2()`.
swaps the byte order of the blob content as it would be an array of `16bits integers`

`swap4()`.
swaps the byte order of the blob content as it would be an array of `32bits integers`

C API

!!FIXME!!

Chapter 4. The Math library

the math lib provides basic mathematic routines. The library mimics the C runtime library implementation.

Squirrel API

Global functions

`sqrt(`*`x`*`)`.
returns the square root of *`x`*

`fabs(`*`x`*`)`.
returns the absolute value of *`x`* as float

`abs(`*`x`*`)`.
returns the absolute value of *`x`* as integer

`sin(`*`x`*`)`.
returns the sine of *`x`*

`cos(`*`x`*`)`.
returns the cosine of *`x`*

`asin(`*`x`*`)`.
returns the arcsine of *`x`*

`acos(`*`x`*`)`.
returns the arccosine of *`x`*

`log(`*`x`*`)`.
returns the natural logarithm of *`x`*

`log10(`*`x`*`)`.
returns the logarithm base-10 of *`x`*

`tan(`*`x`*`)`.
returns the tangent of *`x`*

`atan(`*`x`*`)`.
returns the arctangent of *`x`*

`atan2(`*`x`*`, `*`y`*`)`.
returns the arctangent of *`y/x`*.

`pow(`*`x`*`, `*`y`*`)`.
returns *`x`* raised to the power of *`y`*.

`floor(`*`x`*`)`.
returns a float value representing the largest integer that is less than or equal to *`x`*

`ceil(`*`x`*`)`.
returns a float value representing the smallest integer that is greater than or equal to *`x`*

`exp(`*`x`*`)`.
returns the exponential value of the float parameter *`x`*

`srand(`*`seed`*`)`.
sets the starting point for generating a series of pseudorandom integers

`rand()`.
returns a pseudorandom integer in the range 0 to RAND_MAX

`RAND_MAX`.
the maximum value that can be returned by the `rand()` function

`PI`.
The numeric constant pi (3.141592) is the ratio of the circumference of a circle to its diameter

C API

!!FIXME!!

Chapter 5. The System library

!!FIXME!!

Squirrel API

Global functions

`getenv(`*`varaname`*`).`
Returns a string containing the value of the environment variable *`varname`*

`system(`*`cmd`*`).`
executes the string *`cmd`* through the os command interpreter.

`clock().`
returns a float representing the number of seconds elapsed since the start of the process

`time().`
returns the number of seconds elapsed since midnight 00:00:00, January 1, 1970. the result of this function can be formatted through the faunction `date`

`date(`*`[time]`*`, `*`[format]`*`).`
returns a table containing a date/time splitted in the slots:

`sec`	Seconds after minute (0 – 59).
`min`	Minutes after hour (0 – 59).
`hour`	Hours since midnight (0 – 23).
`day`	Day of month (1 – 31).
`month`	Month (0 – 11; January = 0).
`year`	Year (current year).
`wday`	Day of week (0 – 6; Sunday = 0).
`yday`	Day of year (0 – 365; January 1 = 0).

if *`time`* is omitted the current time is used.
if *`format`* can be 'l' local time or 'u' UTC time, if omitted is defaulted as 'l'(local time).

C API

!!FIXME!!

Chapter 6. The String library

the string lib implements string formatting and manioulation routines.

Squirrel API

Global functions

`format(`*`formatstr`*`, ...)`.
Returns a string formatted according *`formatstr`* and the optional parameters following it. The format string follows the same rules as the `printf` family of standard C functions(the "*" is not supported).

```
eg.
sq> print(format("%s %d 0x%02X\n","this is a test :",123,10));
this is a test : 123 0x0A
```

C API

!!FIXME!!

Chapter 7. The Module library

!!FIXME!!

Squirrel API

Global functions

`import(`*`module`*`).`
loads a module and return a function that initilizes.!!FIXME!!

C API

!!FIXME!!

Index