

Elbonia Client Reference Manual  
0.1

Generated by Doxygen 1.3

Wed May 14 10:53:14 2003



# Contents

<b>1</b>	<b>Elbonia Client Compound Index</b>	<b>1</b>
1.1	Elbonia Client Compound List . . . . .	1
<b>2</b>	<b>Elbonia Client File Index</b>	<b>3</b>
2.1	Elbonia Client File List . . . . .	3
<b>3</b>	<b>Elbonia Client Class Documentation</b>	<b>5</b>
3.1	console_line Struct Reference . . . . .	5
3.2	link Struct Reference . . . . .	6
<b>4</b>	<b>Elbonia Client File Documentation</b>	<b>7</b>
4.1	capabilities.c File Reference . . . . .	7
4.2	capabilities.h File Reference . . . . .	9
4.3	client.c File Reference . . . . .	10
4.4	client.h File Reference . . . . .	12
4.5	console.c File Reference . . . . .	13
4.6	console.h File Reference . . . . .	15
4.7	graphics.c File Reference . . . . .	16
4.8	graphics.h File Reference . . . . .	18
4.9	history.c File Reference . . . . .	19
4.10	history.h File Reference . . . . .	21
4.11	navigation.c File Reference . . . . .	22
4.12	navigation.h File Reference . . . . .	25
4.13	net.c File Reference . . . . .	27
4.14	net.h File Reference . . . . .	30



# Chapter 1

## Elbonia Client Compound Index

### 1.1 Elbonia Client Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

<code>console_line</code> . . . . .	5
<code>link</code> . . . . .	6



## Chapter 2

# Elbonia Client File Index

### 2.1 Elbonia Client File List

Here is a list of all files with brief descriptions:

<b>capabilities.c</b>	7
<b>capabilities.h</b>	9
<b>client.c</b>	10
<b>client.h</b>	12
<b>console.c</b>	13
<b>console.h</b>	15
<b>graphics.c</b>	16
<b>graphics.h</b>	18
<b>history.c</b>	19
<b>history.h</b>	21
<b>navigation.c</b>	22
<b>navigation.h</b>	25
<b>net.c</b>	27
<b>net.h</b>	30





## Chapter 3

# Elbonia Client Class Documentation

### 3.1 console\_line Struct Reference

#### Public Attributes

- char str [40]

#### 3.1.1 Detailed Description

For holding one line of console output.

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 char console\_line::str[40]

The documentation for this struct was generated from the following file:

- console.c

## 3.2 link Struct Reference

```
#include <navigation.h>
```

### Public Attributes

- unsigned int **x1**
- unsigned int **y1**
- unsigned int **x2**
- unsigned int **y2**
- char \* **url**

### 3.2.1 Member Data Documentation

**3.2.1.1 char\* link::url**

**3.2.1.2 unsigned int link::x1**

**3.2.1.3 unsigned int link::x2**

**3.2.1.4 unsigned int link::y1**

**3.2.1.5 unsigned int link::y2**

The documentation for this struct was generated from the following file:

- **navigation.h**

## Chapter 4

# Elbonia Client File Documentation

### 4.1 capabilities.c File Reference

```
#include <vmgp.h>
#include "capabilities.h"
```

#### Functions

- int `get_capabilities` ()

#### Variables

- int `screen_width`
- int `screen_height`
- int `video_modes`

#### 4.1.1 Function Documentation

##### 4.1.1.1 int `get_capabilities` (void)

Get capabilities and set variables defined in **capabilities.h**.

**Returns:**

1 if enough for running Elbonia, else 0.

#### 4.1.2 Variable Documentation

##### 4.1.2.1 int `screen_height`

height of screen in pixels.

**4.1.2.2 int screen\_width**

width of screen in pixels.

**4.1.2.3 int video\_modes**

supported video modes.

---

## 4.2 capabilities.h File Reference

### Functions

- int `get_capabilities` (void)

#### 4.2.1 Function Documentation

##### 4.2.1.1 int `get_capabilities` (void)

Get capabilities and set variables defined in **capabilities.h**.

##### Returns:

1 if enough for running Elbonia, else 0.

## 4.3 client.c File Reference

```
#include <vstream.h>
#include <vmgp.h>
#include <stdlib.h>
#include <string.h>
#include "client.h"
#include "net.h"
#include "navigation.h"
#include "graphics.h"
#include "console.h"
#include "capabilities.h"
#include "history.h"
```

### Functions

- int **get\_page\_information** (void)
- **\_\_attribute\_\_((destructor))** void savestate(void)
- int **main** (void)

### Variables

- uint32\_t **stream**
- char **data** [DATA\_SIZE]
- char \* **image**
- int **data\_size**
- int **page\_height**
- int **page\_width**
- int **scroll\_position**

### 4.3.1 Function Documentation

#### 4.3.1.1 **\_\_attribute\_\_((destructor))**

Called when Elbonia is interrupted (by, for example, an incoming call). We hope this will also be called when the program is forced to quit by the user pressing No for a while. The only place this is documented is in the "Guidelines" provided on the Mophon home page.

#### 4.3.1.2 **int get\_page\_information (void)**

Get page width and height.

#### **Returns:**

non-zero if client is available to display the video mode of the image, 0 otherwise.

**4.3.1.3** `int main (void)`

## **4.3.2** Variable Documentation

**4.3.2.1** `char data[DATA_SIZE]`

received data

**4.3.2.2** `int data_size`

the size of the image and the imagemap

**4.3.2.3** `char* image`

pointer to the image

**4.3.2.4** `int page_height`

the page height

**4.3.2.5** `int page_width`

the page width

**4.3.2.6** `int scroll_position ()`

y position of upper edge of screen

**4.3.2.7** `uint32_t stream`

socket

## 4.4 client.h File Reference

### Defines

- `#define SERVER_ADDRESS "127.0.0.1"`
- `#define SERVER_PORT 4711`
- `#define DATA_SIZE 15000`

### Functions

- `int get_page_information (void)`

### Variables

- `int video_modes`

#### 4.4.1 Define Documentation

4.4.1.1 `#define DATA_SIZE 15000`

4.4.1.2 `#define SERVER_ADDRESS "127.0.0.1"`

4.4.1.3 `#define SERVER_PORT 4711`

#### 4.4.2 Function Documentation

4.4.2.1 `int get_page_information (void)`

Get page width and height.

#### Returns:

non-zero if client is available to display the video mode of the image, 0 otherwise.

#### 4.4.3 Variable Documentation

4.4.3.1 `int video_modes ()`

supported video modes.



## 4.5 console.c File Reference

```
#include <vmgp.h>
#include <stdarg.h>
#include "console.h"
#include "meta.h"
```

### Compounds

- struct `console_line`

### Functions

- void `console_init` (void)
- void `console_render` (void)
- void `print` (char \*str)
- void `println` (char \*str)
- void `print_int` (int n)

### Variables

- VMGPFONT font

#### 4.5.1 Function Documentation

##### 4.5.1.1 void console\_init (void)

Initialize console usage. Must be called before any call to `console_render` or any of the print functions.

##### 4.5.1.2 void console\_render (void)

Render console to screen. This is automatically called after each print call.

##### 4.5.1.3 void print (char \* str)

Print a message to console. Used mainly for debug purposes.

**Parameters:**

*str* Message to print.

##### 4.5.1.4 void print\_int (int n)

Print an integer to console, followed by a newline. Used mainly for debug purposes.

**Parameters:**

*n* Integer to print.

#### 4.5.1.5 void println (char \* *str*)

Print a message to console followed by a newline. Used mainly for debug purposes.

**Parameters:**

*str* Message to print.

### 4.5.2 Variable Documentation

#### 4.5.2.1 VMGPFONT font

**Initial value:**

```
{
    CONSOLE_FONT + CONSOLE_FONT_CHARTBLSIZE,
    CONSOLE_FONT,
    CONSOLE_FONT_BPP,
    CONSOLE_FONT_WIDTH,
    CONSOLE_FONT_HEIGHT,
    CONSOLE_FONT_PALINDEX
}
```

The console font to use.

## 4.6 console.h File Reference

### Functions

- void **console\_init** (void)
- void **console\_render** (void)
- void **print** (char \*str)
- void **println** (char \*str)
- void **print\_int** (int n)

### 4.6.1 Function Documentation

#### 4.6.1.1 void console\_init (void)

Initialize console usage. Must be called before any call to console\_render or any of the print functions.

#### 4.6.1.2 void console\_render (void)

Render console to screen. This is automatically called after each print call.

#### 4.6.1.3 void print (char \* *str*)

Print a message to console. Used mainly for debug purposes.

**Parameters:**

*str* Message to print.

#### 4.6.1.4 void print\_int (int *n*)

Print an integer to console, followed by a newline. Used mainly for debug purposes.

**Parameters:**

*n* Integer to print.

#### 4.6.1.5 void println (char \* *str*)

Print a message to console followed by a newline. Used mainly for debug purposes.

**Parameters:**

*str* Message to print.

## 4.7 graphics.c File Reference

```
#include <stdlib.h>
#include <vmgp.h>
#include "navigation.h"
#include "graphics.h"
```

### Functions

- int **render** (const char \***image**)
- void **get\_and\_set\_palette** (const char \***image**, char \*\***rest**)

### Variables

- link **imagemap** [NUM\_LINKS]
- int **scroll\_position**
- int **link\_active**
- int **page\_width**
- int **page\_height**
- int **screen\_width**
- int **screen\_height**

#### 4.7.1 Function Documentation

##### 4.7.1.1 void **get\_and\_set\_palette** (const char \* *image*, char \*\* *rest*)

Used to extract palette information when indexed palette images are sent to the client.

##### Parameters:

- *image* The image data
- *rest* Pointer into image indicating start of image data

##### 4.7.1.2 int **render** (const char \* *image*)

Render screen and everything on it.

##### Parameters:

- *image* The image to draw. Should point to a SPRITE struct, as defined by the Mophun API reference.

##### Returns:

- 1 on success, 0 otherwise.

#### 4.7.2 Variable Documentation

##### 4.7.2.1 link **imagemap**[NUM\_LINKS] ()

the links

**4.7.2.2 int link\_active ()**

active link

**4.7.2.3 int page\_height ()**

the page height

**4.7.2.4 int page\_width ()**

the page width

**4.7.2.5 int screen\_height ()**

height of screen in pixels.

**4.7.2.6 int screen\_width ()**

width of screen in pixels.

**4.7.2.7 int scroll\_position ()**

y position of upper edge of screen

## 4.8 graphics.h File Reference

### Functions

- int **render** (const char \*)
- void **get\_and\_set\_palette** (const char \*, char \*\*)

### Variables

- link **imagemap** [NUM.LINKS]

### 4.8.1 Function Documentation

#### 4.8.1.1 void **get\_and\_set\_palette** (const char \* *image*, char \*\* *rest*)

Used to extract palette information when indexed palette images are sent to the client.

**Parameters:**

- image* The image data
- rest* Pointer into image indicating start of image data

#### 4.8.1.2 int **render** (const char \* *image*)

Render screen and everything on it.

**Parameters:**

- image* The image to draw. Should point to a SPRITE struct, as defined by the Mophun API reference.

**Returns:**

- 1 on success, 0 otherwise.

### 4.8.2 Variable Documentation

#### 4.8.2.1 link **imagemap**[NUM.LINKS] ()

the links

## 4.9 history.c File Reference

```
#include <string.h>
#include "history.h"
```

### Functions

- void **history\_add** (const char \*page)
- char \* **history\_get** (void)
- int **history\_is\_empty** (void)

### Variables

- char **history\_item** [NUM\_HISTORY\_ITEMS][HISTORY\_ITEM\_LENGTH]
- int **first** = -1
- int **next** = 0

### 4.9.1 Function Documentation

#### 4.9.1.1 void history\_add (const char \* page)

Add page to history.

**Parameters:**

*page* The page to add to history.

#### 4.9.1.2 char\* history\_get (void)

Get most recently visited page. The behaviour if history is empty is undefined.

**Returns:**

The address of the most recently visited page.

#### 4.9.1.3 int history\_is\_empty (void)

**Returns:**

1 if history is empty, 0 if not.

### 4.9.2 Variable Documentation

#### 4.9.2.1 int first = -1

first item in list. -1 means empty.

#### 4.9.2.2 char history\_item[NUM\_HISTORY\_ITEMS][HISTORY\_ITEM\_LENGTH]

all the history items currently stored.

**4.9.2.3 int next = 0**

next free slot for items.



## 4.10 history.h File Reference

### Defines

- #define **NUM\_HISTORY\_ITEMS** 5
- #define **HISTORY\_ITEM\_LENGTH** 256

### Functions

- void **history\_add** (const char \*)
- char \* **history\_get** (void)
- int **history\_is\_empty** (void)

#### 4.10.1 Define Documentation

4.10.1.1 #define **HISTORY\_ITEM\_LENGTH** 256

4.10.1.2 #define **NUM\_HISTORY\_ITEMS** 5

#### 4.10.2 Function Documentation

4.10.2.1 void **history\_add** (const char \* *page*)

Add page to history.

**Parameters:**

*page* The page to add to history.

4.10.2.2 char\* **history\_get** (void)

Get most recently visited page. The behaviour if history is empty is undefined.

**Returns:**

The address of the most recently visited page.

4.10.2.3 int **history\_is\_empty** (void)

**Returns:**

1 if history is empty, 0 if not.

## 4.11 navigation.c File Reference

```
#include <string.h>
#include "navigation.h"
#include "client.h"
#include "net.h"
#include "history.h"
```

### Functions

- int **link\_follow** (int **stream**)
- int **link\_is\_visible** (int **link\_index**)
- int **next\_link** (void)
- int **previous\_link** (void)
- int **scroll\_up** (void)
- int **scroll\_down** (void)
- int **get\_imagemap** (const char \***data**, char \*\***rest**)

### Variables

- int **num\_links**
- link **imagemap** [NUM\_LINKS]
- int **link\_active**
- int **scroll\_position**
- int **scroll\_speed** = 4
- int **screen\_height**
- int **page\_height**
- char **data** [DATA\_SIZE]
- int **data\_size**

### 4.11.1 Function Documentation

#### 4.11.1.1 int **get\_imagemap** (const char \* *data*, char \*\* *rest*)

Extract imagemap from data. Imagemap looks like (everything in network byte order)

- Number of links (4 bytes)
- Each link:
  - x1 (4 bytes)
  - y1 (4 bytes)
  - x2 (4 bytes)
  - y2 (4 bytes)
  - path (null-terminated)

If new page contains any links, the first one is set as active link. Otherwise no link becomes active.

**Parameters:**

*data* Data to extract imagemap from.

*rest* Set to point to byte immediately following imagemap.

**Returns:**

Number of hotspots in imagemap

**4.11.1.2 int link\_follow (int stream)**

Follow active link if it's visible. If a link is followed, the page is set to be viewed from top and the path to the current page is saved.

**Parameters:**

*stream* Socket to which request should be sent.

**Returns:**

1 on success, else 0.

**4.11.1.3 int link\_is\_visible (int link\_index)**

Determine whether a link is entirely visible.

**Parameters:**

*link\_index* Index of link.

**Returns:**

1 if entire link is visible, -1 if link doesn't exist and 0 otherwise.

**4.11.1.4 int next\_link (void)**

Move to next visible link (if one is present).

**Returns:**

New active link.

**4.11.1.5 int previous\_link (void)**

Move to previous visible link (if one is present).

**Returns:**

New active link.

**4.11.1.6 int scroll\_down (void)**

Scroll down. If active link becomes invisible and next link is visible it becomes the active link. If no link is visible active link is unchanged.

**Returns:**

New active link.

**4.11.1.7 int scroll\_up (void)**

Scroll up. If active link becomes invisible and previous link is visible it becomes the active link. If no link is visible active link is unchanged.

**Returns:**

New active link.

**4.11.2 Variable Documentation****4.11.2.1 char data[DATA\_SIZE] ()**

received data

**4.11.2.2 int data\_size ()**

the size of the image and the imagemap

**4.11.2.3 link imagemap[NUM\_LINKS]**

the links

**4.11.2.4 int link\_active**

active link

**4.11.2.5 int num\_links**

number of links on the page

**4.11.2.6 int page\_height ()**

the page height

**4.11.2.7 int screen\_height ()**

height of screen in pixels.

**4.11.2.8 int scroll\_position**

y position of upper edge of screen

**4.11.2.9 int scroll\_speed = 4**

speed of scrolling

## 4.12 navigation.h File Reference

### Compounds

- struct **link**

### Defines

- `#define NUM_LINKS 20`

### Functions

- int **link\_follow** (int)
- int **link\_is\_visible** (int)
- int **next\_link** (void)
- int **previous\_link** (void)
- int **scroll\_up** (void)
- int **scroll\_down** (void)
- int **get\_imagemap** (const char \*, char \*\*)

#### 4.12.1 Define Documentation

##### 4.12.1.1 `#define NUM_LINKS 20`

#### 4.12.2 Function Documentation

##### 4.12.2.1 int **get\_imagemap** (const char \* *data*, char \*\* *rest*)

Extract imagemap from data. Imagemap looks like (everything in network byte order)

- Number of links (4 bytes)
- Each link:
  - x1 (4 bytes)
  - y1 (4 bytes)
  - x2 (4 bytes)
  - y2 (4 bytes)
  - path (null-terminated)

If new page contains any links, the first one is set as active link. Otherwise no link becomes active.

#### Parameters:

*data* Data to extract imagemap from.

*rest* Set to point to byte immediately following imagemap.

#### Returns:

Number of hotspots in imagemap

#### 4.12.2.2 `int link_follow (int stream)`

Follow active link if it's visible. If a link is followed, the page is set to be viewed from top and the path to the current page is saved.

**Parameters:**

*stream* Socket to which request should be sent.

**Returns:**

1 on success, else 0.

#### 4.12.2.3 `int link_is_visible (int link_index)`

Determine whether a link is entirely visible.

**Parameters:**

*link\_index* Index of link.

**Returns:**

1 if entire link is visible, -1 if link doesn't exist and 0 otherwise.

#### 4.12.2.4 `int next_link (void)`

Move to next visible link (if one is present).

**Returns:**

New active link.

#### 4.12.2.5 `int previous_link (void)`

Move to previous visible link (if one is present).

**Returns:**

New active link.

#### 4.12.2.6 `int scroll_down (void)`

Scroll down. If active link becomes invisible and next link is visible it becomes the active link. If no link is visible active link is unchanged.

**Returns:**

New active link.

#### 4.12.2.7 `int scroll_up (void)`

Scroll up. If active link becomes invisible and previous link is visible it becomes the active link. If no link is visible active link is unchanged.

**Returns:**

New active link.

## 4.13 net.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <vstream.h>
#include "net.h"
#include "console.h"
```

### Functions

- int **connect** (const char \*address, int port)
- int **request** (uint32\_t **stream**, const char \*location, void \*buffer, uint32\_t size)
- int **write** (uint32\_t **stream**, void \*buffer, uint32\_t size)
- int **read** (uint32\_t **stream**, void \*buffer, uint32\_t size)
- uint16\_t **net\_to\_hosts** (uint16\_t input)
- uint32\_t **net\_to\_hosti** (uint32\_t input)

### Variables

- int **screen\_height**
- int **screen\_width**

#### 4.13.1 Function Documentation

##### 4.13.1.1 int connect (const char \* *address*, int *port*)

Connect to elbonia server at address:port.

**Parameters:**

*address* Remote address.

*port* Remote port.

**Returns:**

The connection socket or -1 if failed.

##### 4.13.1.2 uint32\_t net\_to\_hosti (uint32\_t *input*)

Convert network (big endian) to host order.

**Parameters:**

*input* The int to be converted to host order

**Returns:**

input converted to host order

#### 4.13.1.3 `uint16_t net_to_hosts (uint16_t input)`

Convert network (big endian) to host order.

**Parameters:**

*input* The short to be converted to host order

**Returns:**

input converted to host order

#### 4.13.1.4 `int read (uint32_t stream, void * buffer, uint32_t size)`

Read size number of bytes from stream into buffer.

**Parameters:**

*stream* Socket to read from.

*buffer* Buffer to fill with read data.

*size* Size of buffer.

**Returns:**

number of bytes read. -1 on error, 0 on end of file.

#### 4.13.1.5 `int request (uint32_t stream, const char * location, void * buffer, uint32_t size)`

Request a page from stream. buffer is filled with response. size indicates size of buffer. TODO: this is not at all done.

**Parameters:**

*stream* Socket to write to.

*location* Identifier of resource to request.

*buffer* Buffer to fill with response.

*size* Size of buffer.

**Returns:**

1 on success, 0 on error.

#### 4.13.1.6 `int write (uint32_t stream, void * buffer, uint32_t size)`

Write size number of bytes from buffer to stream.

**Parameters:**

*stream* Socket to write to.

*buffer* Buffer to read from.

*size* Size of buffer.

**Returns:**

Number of bytes written, -1 on error.



## 4.13.2 Variable Documentation

### 4.13.2.1 int screen\_height ()

height of screen in pixels.

### 4.13.2.2 int screen\_width ()

width of screen in pixels.

## 4.14 net.h File Reference

```
#include <vmgp.h>
```

### Functions

- int **connect** (const char \*, int)
- int **request** (uint32\_t, const char \*, void \*, uint32\_t)
- int **write** (uint32\_t, void \*, uint32\_t)
- int **read** (uint32\_t, void \*, uint32\_t)
- uint16\_t **net\_to\_hosts** (uint16\_t)
- uint32\_t **net\_to\_hosti** (uint32\_t)

### 4.14.1 Function Documentation

#### 4.14.1.1 int connect (const char \* *address*, int *port*)

Connect to elbonia server at address:port.

**Parameters:**

*address* Remote address.

*port* Remote port.

**Returns:**

The connection socket or -1 if failed.

#### 4.14.1.2 uint32\_t net\_to\_hosti (uint32\_t *input*)

Convert network (big endian) to host order.

**Parameters:**

*input* The int to be converted to host order

**Returns:**

input converted to host order

#### 4.14.1.3 uint16\_t net\_to\_hosts (uint16\_t *input*)

Convert network (big endian) to host order.

**Parameters:**

*input* The short to be converted to host order

**Returns:**

input converted to host order

#### 4.14.1.4 int read (uint32\_t *stream*, void \* *buffer*, uint32\_t *size*)

Read size number of bytes from stream into buffer.

**Parameters:**

*stream* Socket to read from.

*buffer* Buffer to fill with read data.

*size* Size of buffer.

**Returns:**

number of bytes read. -1 on error, 0 on end of file.

#### 4.14.1.5 int request (uint32\_t *stream*, const char \* *location*, void \* *buffer*, uint32\_t *size*)

Request a page from stream. *buffer* is filled with response. *size* indicates size of buffer. TODO: this is not at all done.

**Parameters:**

*stream* Socket to write to.

*location* Identifier of resource to request.

*buffer* Buffer to fill with response.

*size* Size of buffer.

**Returns:**

1 on success, 0 on error.

#### 4.14.1.6 int write (uint32\_t *stream*, void \* *buffer*, uint32\_t *size*)

Write size number of bytes from buffer to stream.

**Parameters:**

*stream* Socket to write to.

*buffer* Buffer to read from.

*size* Size of buffer.

**Returns:**

Number of bytes written, -1 on error.

# Index

- `__attribute__`
  - `client.c`, 10
- `capabilities.c`, 7
  - `get_capabilities`, 7
  - `screen_height`, 7
  - `screen_width`, 7
  - `video_modes`, 8
- `capabilities.h`, 9
  - `get_capabilities`, 9
- `client.c`, 10
  - `__attribute__`, 10
  - `data`, 11
  - `data_size`, 11
  - `get_page_information`, 10
  - `image`, 11
  - `main`, 10
  - `page_height`, 11
  - `page_width`, 11
  - `scroll_position`, 11
  - `stream`, 11
- `client.h`, 12
  - `DATA_SIZE`, 12
  - `get_page_information`, 12
  - `SERVER_ADDRESS`, 12
  - `SERVER_PORT`, 12
  - `video_modes`, 12
- `connect`
  - `net.c`, 27
  - `net.h`, 30
- `console.c`, 13
  - `console_init`, 13
  - `console_render`, 13
  - `font`, 14
  - `print`, 13
  - `print_int`, 13
  - `println`, 13
- `console.h`, 15
  - `console_init`, 15
  - `console_render`, 15
  - `print`, 15
  - `print_int`, 15
  - `println`, 15
- `console_init`
  - `console.c`, 13
  - `console.h`, 15
- `console_line`
  - `str`, 5
- `console_render`
  - `console.c`, 13
  - `console.h`, 15
- `data`
  - `client.c`, 11
  - `navigation.c`, 24
- `DATA_SIZE`
  - `client.h`, 12
- `data_size`
  - `client.c`, 11
  - `navigation.c`, 24
- `first`
  - `history.c`, 19
- `font`
  - `console.c`, 14
- `get_and_set_palette`
  - `graphics.c`, 16
  - `graphics.h`, 18
- `get_capabilities`
  - `capabilities.c`, 7
  - `capabilities.h`, 9
- `get_imagemap`
  - `navigation.c`, 22
  - `navigation.h`, 25
- `get_page_information`
  - `client.c`, 10
  - `client.h`, 12
- `graphics.c`, 16
  - `get_and_set_palette`, 16
  - `imagemap`, 16
  - `link_active`, 16
  - `page_height`, 17
  - `page_width`, 17
  - `render`, 16
  - `screen_height`, 17
  - `screen_width`, 17
  - `scroll_position`, 17
- `graphics.h`, 18
  - `get_and_set_palette`, 18

- imagemap, 18
  - render, 18
- history.c, 19
  - first, 19
  - history\_add, 19
  - history\_get, 19
  - history\_is\_empty, 19
  - history\_item, 19
  - next, 19
- history.h, 21
  - history\_add, 21
  - history\_get, 21
  - history\_is\_empty, 21
  - HISTORY\_ITEM\_LENGTH, 21
  - NUM\_HISTORY\_ITEMS, 21
- history\_add
  - history.c, 19
  - history.h, 21
- history\_get
  - history.c, 19
  - history.h, 21
- history\_is\_empty
  - history.c, 19
  - history.h, 21
- history\_item
  - history.c, 19
- HISTORY\_ITEM\_LENGTH
  - history.h, 21
- image
  - client.c, 11
- imagemap
  - graphics.c, 16
  - graphics.h, 18
  - navigation.c, 24
- link, 6
  - url, 6
  - x1, 6
  - x2, 6
  - y1, 6
  - y2, 6
- link\_active
  - graphics.c, 16
  - navigation.c, 24
- link\_follow
  - navigation.c, 23
  - navigation.h, 25
- link\_is\_visible
  - navigation.c, 23
  - navigation.h, 26
- main
  - client.c, 10
- navigation.c, 22
  - data, 24
  - data\_size, 24
  - get\_imagemap, 22
  - imagemap, 24
  - link\_active, 24
  - link\_follow, 23
  - link\_is\_visible, 23
  - next\_link, 23
  - num\_links, 24
  - page\_height, 24
  - previous\_link, 23
  - screen\_height, 24
  - scroll\_down, 23
  - scroll\_position, 24
  - scroll\_speed, 24
  - scroll\_up, 23
- navigation.h, 25
  - get\_imagemap, 25
  - link\_follow, 25
  - link\_is\_visible, 26
  - next\_link, 26
  - NUM\_LINKS, 25
  - previous\_link, 26
  - scroll\_down, 26
  - scroll\_up, 26
- net.c, 27
  - connect, 27
  - net\_to\_hosti, 27
  - net\_to\_hosts, 27
  - read, 28
  - request, 28
  - screen\_height, 29
  - screen\_width, 29
  - write, 28
- net.h, 30
  - connect, 30
  - net\_to\_hosti, 30
  - net\_to\_hosts, 30
  - read, 30
  - request, 31
  - write, 31
- net\_to\_hosti
  - net.c, 27
  - net.h, 30
- net\_to\_hosts
  - net.c, 27
  - net.h, 30
- next
  - history.c, 19
- next\_link
  - navigation.c, 23

---

- navigation.h, 26
- NUM\_HISTORY\_ITEMS
  - history.h, 21
- NUM\_LINKS
  - navigation.h, 25
- num\_links
  - navigation.c, 24
- page\_height
  - client.c, 11
  - graphics.c, 17
  - navigation.c, 24
- page\_width
  - client.c, 11
  - graphics.c, 17
- previous\_link
  - navigation.c, 23
  - navigation.h, 26
- print
  - console.c, 13
  - console.h, 15
- print\_int
  - console.c, 13
  - console.h, 15
- println
  - console.c, 13
  - console.h, 15
- read
  - net.c, 28
  - net.h, 30
- render
  - graphics.c, 16
  - graphics.h, 18
- request
  - net.c, 28
  - net.h, 31
- screen\_height
  - capabilities.c, 7
  - graphics.c, 17
  - navigation.c, 24
  - net.c, 29
- screen\_width
  - capabilities.c, 7
  - graphics.c, 17
  - net.c, 29
- scroll\_down
  - navigation.c, 23
  - navigation.h, 26
- scroll\_position
  - client.c, 11
  - graphics.c, 17
  - navigation.c, 24
- scroll\_speed
  - navigation.c, 24
- scroll\_up
  - navigation.c, 23
  - navigation.h, 26
- SERVER\_ADDRESS
  - client.h, 12
- SERVER\_PORT
  - client.h, 12
- str
  - console\_line, 5
- stream
  - client.c, 11
- url
  - link, 6
- video\_modes
  - capabilities.c, 8
  - client.h, 12
- write
  - net.c, 28
  - net.h, 31
- x1
  - link, 6
- x2
  - link, 6
- y1
  - link, 6
- y2
  - link, 6