Elbonia Server Reference Manual

0.1

Generated by Doxygen 1.3

Wed May 14 10:55:09 2003

Contents

Chapter 1

Elbonia Server Compound Index

1.1 Elbonia Server Compound List

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

Chapter 2

Elbonia Server File Index

2.1 Elbonia Server File List

Here is a list of all files with brief descriptions:

Chapter 3

Elbonia Server Class Documentation

3.1 option Struct Reference

`#include <getopt.h>`

Public Attributes

- char * **name**
- int **has_arg**
- int * **flag**
- int **val**

3.1.1 Member Data Documentation

3.1.1.1 int* option::flag

3.1.1.2 int option::has_arg

3.1.1.3 char* option::name

3.1.1.4 int option::val

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

- **getopt.h**

Chapter 4

Elbonia Server File Documentation

4.1 commandline.c File Reference

```
#include "common.h"
#include "error.h"
#include "getopt.h"
#include "commandline.h"
```

Functions

- int **parse_command_line** (int argc, char ******argv**)

Variables

- int **server_port**
- char ***** **server_root**
- int **reverse_lookup**
- int **run_as_daemon**

4.1.1 Function Documentation

4.1.1.1 int parse_command_line (int *argc*, char ** *argv*)

Parse command line. Set global variables to reflect options. FIXME: atoi doesn't detect errors.

Parameters:

argc Number of command line arguments.

argv Command line arguments.

Returns:

1 on success, 0 otherwise.

4.1.2 Variable Documentation

4.1.2.1 int reverse_lookup ()

whether to do reverse lookups.

4.1.2.2 int run_as_daemon ()

whether to run as daemon.

4.1.2.3 int server_port ()

port for listening for connections.

4.1.2.4 char* server_root ()

server root directory.

4.2 commandline.h File Reference

Functions

- int **parse_command_line** (int, char **)

4.2.1 Function Documentation

4.2.1.1 int parse_command_line (int *argc*, char ** *argv*)

Parse command line. Set global variables to reflect options. FIXME: atoi doesn't detect errors.

Parameters:

argc Number of command line arguments.

argv Command line arguments.

Returns:

1 on success, 0 otherwise.

4.3 common.h File Reference

```
#include <stdio.h>
#include <sys/types.h>
```

Defines

- #define **basename**(path) path
- #define **BEGIN_C_DECLS**
- #define **END_C_DECLS**
- #define **XCALLOC**(type, num) ((type *) xcalloc ((num), sizeof(type)))
- #define **XMALLOC**(type, num) ((type *) xmalloc ((num) * sizeof(type)))
- #define **XREALLOC**(type, p, num) ((type *) xrealloc ((p), (num) * sizeof(type)))
- #define **XFREE**(stale)
- #define **PACKAGE_NAME** "Elbonia Server"
- #define **PACKAGE_VERSION** "(No version info available)"
- #define **PACKAGE_STRING** PACKAGE_NAME " " PACKAGE_VERSION
- #define **PACKAGE_TITLE** PACKAGE_STRING
- #define **EXIT_SUCCESS** 0
- #define **EXIT_FAILURE** 1

Functions

- BEGIN_C_DECLS void * **xcalloc** (size_t num, size_t size)
- void * **xmalloc** (size_t num)
- void * **xrealloc** (void *p, size_t num)
- char * **xstrdup** (const char *string)

Variables

- int **errno**
- int **h_errno**

4.3.1 Define Documentation

4.3.1.1 #define basename(path) path

4.3.1.2 #define BEGIN_C_DECLS

4.3.1.3 #define END_C_DECLS

4.3.1.4 #define EXIT_FAILURE 1

4.3.1.5 #define EXIT_SUCCESS 0

4.3.1.6 #define PACKAGE_NAME "Elbonia Server"

4.3.1.7 #define PACKAGE_STRING PACKAGE_NAME " " PACKAGE_VERSION

4.3.1.8 #define PACKAGE_TITLE PACKAGE_STRING

4.3.1.9 #define PACKAGE_VERSION "(No version info available)"

4.3.1.10 #define XCALLOC(type, num) ((type *) xcalloc ((num), sizeof(type)))

4.3.1.11 #define XFREE(stale)

Value:

```
do {        \
       if (stale) { free (stale);  stale = 0; }            \
                                              } while (0)
```

4.3.1.12 #define XMALLOC(type, num) ((type *) xmalloc ((num) * sizeof(type)))

4.3.1.13 #define XREALLOC(type, p, num) ((type *) xrealloc ((p), (num) * sizeof(type)))

4.3.2 Function Documentation

4.3.2.1 BEGIN_C_DECLS void* xcalloc (size_t *num*, size_t *size*)

Allocate and zero memory with sanity checking. Raises a fatal error if memory can't be allocated.

Parameters:

num Number of elements to allocate.

size Size of each element.

4.3.2.2 void* xmalloc (size_t *num*)

Allocate memory with sanity checking. Raises a fatal error if memory can't be allocated.

Parameters:
num Number of bytes to allocate.

Returns:
Pointer to allocated memory.

4.3.2.3 void* xrealloc (void * *p*, size_t *num*)

Reallocate memory with sanity checking. Raises a fatal error if memory can't be reallocated.

Parameters:
p Pointer to already allocated memory.

num Number bytes to allocate.

Returns:
Pointer to allocated memory.

4.3.2.4 char* xstrdup (const char * *string*)

Duplicate a string. Raises fatal error if memory for the string can't be allocated.

Parameters:
string String to duplicate.

Returns:
A copy of the string.

4.3.3 Variable Documentation

4.3.3.1 int errno

4.3.3.2 int h_errno

4.4 error.c File Reference

```
#include "common.h"
#include "error.h"
```

Functions

- void **set_program_name** (const char *path)
- void **error_warning** (const char *message)
- void **error_error** (const char *message)
- void **error_fatal** (const char *message)

Variables

- const char * **program_name** = NULL

4.4.1 Function Documentation

4.4.1.1 void error_error (const char * *message*)

Raise an error message.

Parameters:
message Error message.

4.4.1.2 void error_fatal (const char * *message*)

Raise a fatal error message.

Parameters:
message Error message.

4.4.1.3 void error_warning (const char * *message*)

Raise a warning message.

Parameters:
message Warning message.

4.4.1.4 void set_program_name (const char * *path*)

Set name of program. Used by error reporting routines. If given name is a path, set name to basename of the path.

Parameters:
path Path to, or name of, program.

4.4.2 Variable Documentation

4.4.2.1 const char* program_name = NULL

program name used by error routines.

4.5 error.h File Reference

Functions

- void **set_program_name** (const char *argv0)
- void **error_warning** (const char *message)
- void **error_error** (const char *message)
- void **error_fatal** (const char *message)

Variables

- BEGIN_C_DECLS const char * **program_name**

4.5.1 Function Documentation

4.5.1.1 void error_error (const char * *message*)

Raise an error message.

Parameters:
message Error message.

4.5.1.2 void error_fatal (const char * *message*)

Raise a fatal error message.

Parameters:
message Error message.

4.5.1.3 void error_warning (const char * *message*)

Raise a warning message.

Parameters:
message Warning message.

4.5.1.4 void set_program_name (const char * *path*)

Set name of program. Used by error reporting routines. If given name is a path, set name to basename of the path.

Parameters:
path Path to, or name of, program.

4.5.2 Variable Documentation

4.5.2.1 BEGIN_C_DECLS const char* program_name ()

program name used by error routines.

4.6 getopt.c File Reference

```
#include <strings.h>
#include <stdio.h>
#include "getopt.h"
```

Defines

- #define **GETOPT_INTERFACE_VERSION** 2
- #define **_**(msgid) (msgid)
- #define **SWAP_FLAGS**(ch1, ch2)
- #define **NONOPTION_P** (**argv**[**optind**][0] != '-' || **argv**[**optind**][1] == '\0')

Enumerations

- enum { **REQUIRE_ORDER**, **PERMUTE**, **RETURN_IN_ORDER** }

Functions

- char * **getenv** ()

Variables

- char * **optarg** = NULL
- int **optind** = 1
- int **__getopt_initialized** = 0
- int **opterr** = 1
- int **optopt** = '?'
- int **chr**
- char *const * **argv**
- const char * **optstring**
- const struct **option** * **longopts**
- int * **longind**
- int **long_only**

4.6.1 Define Documentation

4.6.1.1 #define _(msgid) (msgid)

4.6.1.2 #define GETOPT_INTERFACE_VERSION 2

4.6.1.3 #define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0')

4.6.1.4 #define SWAP_FLAGS(ch1, ch2)

4.6.2 Enumeration Type Documentation

4.6.2.1 anonymous enum

Enumeration values:

REQUIRE_ORDER

PERMUTE

RETURN_IN_ORDER

4.6.3 Function Documentation

4.6.3.1 char* getenv ()

4.6.4 Variable Documentation

4.6.4.1 int __getopt_initialized = 0

4.6.4.2 char *const * argv

4.6.4.3 int chr

4.6.4.4 int long_only

4.6.4.5 int* longind

4.6.4.6 const struct option* longopts

4.6.4.7 char* optarg = NULL

4.6.4.8 int opterr = 1

4.6.4.9 int optind = 1

4.6.4.10 int optopt = '?'

4.6.4.11 const char * optstring

4.7 getopt.h File Reference

Compounds

- struct **option**

Defines

- #define **_GETOPT_H** 1
- #define **no_argument** 0
- #define **required_argument** 1
- #define **optional_argument** 2

Functions

- int **getopt** ()
- int **getopt_long** ()
- int **getopt_long_only** ()
- int **_getopt_internal** ()

Variables

- char * **optarg**
- int **optind**
- int **opterr**
- int **optopt**

4.7.1 Define Documentation

4.7.1.1 #define _GETOPT_H 1

4.7.1.2 #define no_argument 0

4.7.1.3 #define optional_argument 2

4.7.1.4 #define required_argument 1

4.7.2 Function Documentation

4.7.2.1 int _getopt_internal ()

4.7.2.2 int getopt ()

4.7.2.3 int getopt_long ()

4.7.2.4 int getopt_long_only ()

4.7.3 Variable Documentation

4.7.3.1 char* optarg

4.7.3.2 int opterr

4.7.3.3 int optind

4.7.3.4 int optopt

4.8 getopt1.c File Reference

```
#include "getopt.h"
#include <stdio.h>
```

Defines

- #define **GETOPT_INTERFACE_VERSION** 2
- #define **NULL** 0

Functions

- int **getopt_long** (argc, **argv**, **options**, **long_options**, **opt_index**) int argc

Variables

- char *const * **argv**
- const char * **options**
- const struct **option** * **long_options**
- int * **opt_index**

4.8.1 Define Documentation

4.8.1.1 #define GETOPT_INTERFACE_VERSION 2

4.8.1.2 #define NULL 0

4.8.2 Function Documentation

4.8.2.1 int getopt_long (argc, argv, options, long_options, opt_index)

4.8.3 Variable Documentation

4.8.3.1 char* const* argv

4.8.3.2 const struct option * long_options

4.8.3.3 int * opt_index

4.8.3.4 const char * options

4.9 net.c File Reference

```
#include "common.h"
```

```
#include "net.h"
```

Functions

- ssize_t **slow_send** (int socket, const void *msg, size_t len, int flags)
- char * **get_request** (int socket)
- int **respond** (int socket, char *request)
- int **response** (int socket, int resp, const char *uri, int width, int height)
- int **response_error** (int socket, int resp)
- char * **response_description** (int response)
- char * **get_hostname** (struct sockaddr_in *address)
- int **istoken** (char *string)
- int **isseparator** (char c)

Variables

- int **reverse_lookup**

4.9.1 Function Documentation

4.9.1.1 char* get_hostname (struct sockaddr_in * *address*)

Parameters:
address Address of remote host.

Returns:
host name of remote address. If reverse lookup is enabled, return hostname, or if something went wrong while doing reverse lookup, return ip address, else return ip address.

4.9.1.2 char* get_request (int *socket*)

Get request from socket.

Parameters:
socket The socket to read from.

Returns:
The given request.

4.9.1.3 int isseparator (char *c*)

Parameters:
c Character to check.

Returns:
1 if char is HTTP "separator" [RFC 2616], 0 otherwise.

4.9.1.4 int istoken (char * *string*)

Parameters:
string String to check.

Returns:
1 if string starts with an HTTP "token" [RFC 2616], 0 otherwise.

4.9.1.5 int respond (int *socket*, char * *request*)

Respond to request on socket. The request parameter may be altered. TODO : should also handle method HEAD FIXME: this function is way to generous with request syntax. also, atoi doesn't check for errors.

Parameters:
socket The socket to write on.

request The request to respond to.

Returns:
0 on failure, 1 otherwise.

4.9.1.6 int response (int *socket*, int *resp*, const char * *uri*, int *width*, int *height*)

Send response to socket. If response is ETP_RESPONSE_OK, also send contents of uri.

First looks for file called -, then .

FIXME: normal url syntax (NN should be interpreted as 0xNN) should be understood.

Parameters:
socket The socket to write to.

resp The reponse code.

uri The requested resource.

width The screen width given by the client.

height The screen height given by the client.

Returns:
1 on success, 0 on failure (ETP protocol errors do not count as failure).

4.9.1.7 char* response_description (int *response*)

Returns:
A string describing a response or NULL if not available.

4.9.1.8 int response_error (int *socket*, int *resp*)

Send an error response of type resp.

Parameters:

socket The socket to write to.

resp The response error code.

Returns:

1 on success, 0 otherwise.

4.9.1.9 ssize_t slow_send (int *socket*, const void * *msg*, size_t *len*, int *flags*)

Send 200 bytes of data, wait 0.3 s and repeat. This function is only used when ERICSSON_T300_COMPATIBILITY_MODE is defined.

Parameters:

socket The socket on which to send data.

msg The data to send.

len Size of data to send.

flags Flags to give to send(3).

Returns:

number of bytes sent, -1 if failed.

4.9.2 Variable Documentation

4.9.2.1 int reverse_lookup ()

whether to do reverse lookups.

4.10 net.h File Reference

Defines

- #define **ETP_VERSION** "ETP/1.0"
- #define **ETP_METHOD_GET** "GET"
- #define **ETP_RESPONSE_OK** 200
- #define **ETP_RESPONSE_BAD_REQUEST** 400
- #define **ETP_RESPONSE_FORBIDDEN** 403
- #define **ETP_RESPONSE_NOT_FOUND** 404
- #define **ETP_RESPONSE_INTERNAL_SERVER_ERROR** 500
- #define **ETP_RESPONSE_NOT_IMPLEMENTED** 501
- #define **dispatch**(socket, buf, len, flags) send(socket, buf, len, flags)

Functions

- ssize_t **slow_send** (int, const void *, size_t, int)
- char * **get_request** (int)
- int **respond** (int, char *)
- int **response** (int, int, const char *, int, int)
- int **response_error** (int, int)
- char * **response_description** (int)
- char * **get_hostname** (struct sockaddr_in *)
- int **istoken** (char *)
- int **isseparator** (char c)

4.10.1 Define Documentation

4.10.1.1 #define dispatch(socket, buf, len, flags) send(socket, buf, len, flags)

4.10.1.2 #define ETP_METHOD_GET "GET"

4.10.1.3 #define ETP_RESPONSE_BAD_REQUEST 400

4.10.1.4 #define ETP_RESPONSE_FORBIDDEN 403

4.10.1.5 #define ETP_RESPONSE_INTERNAL_SERVER_ERROR 500

4.10.1.6 #define ETP_RESPONSE_NOT_FOUND 404

4.10.1.7 #define ETP_RESPONSE_NOT_IMPLEMENTED 501

4.10.1.8 #define ETP_RESPONSE_OK 200

4.10.1.9 #define ETP_VERSION "ETP/1.0"

4.10.2 Function Documentation

4.10.2.1 char* get_hostname (struct sockaddr_in * *address*)

Parameters:

address Address of remote host.

Returns:
host name of remote address. If reverse lookup is enabled, return hostname, or if something went wrong while doing reverse lookup, return ip address, else return ip address.

4.10.2.2 char* get_request (int *socket*)

Get request from socket.

Parameters:
socket The socket to read from.

Returns:
The given request.

4.10.2.3 int isseparator (char *c*)

Parameters:
c Character to check.

Returns:
1 if char is HTTP "separator" [RFC 2616], 0 otherwise.

4.10.2.4 int istoken (char * *string*)

Parameters:
string String to check.

Returns:
1 if string starts with an HTTP "token" [RFC 2616], 0 otherwise.

4.10.2.5 int respond (int *socket*, char * *request*)

Respond to request on socket. The request parameter may be altered. TODO : should also handle method HEAD FIXME: this function is way to generous with request syntax. also, atoi doesn't check for errors.

Parameters:
socket The socket to write on.

request The request to respond to.

Returns:
0 on failure, 1 otherwise.

4.10.2.6 int response (int *socket*, int *resp*, const char * *uri*, int *width*, int *height*)

Send response to socket. If response is ETP_RESPONSE_OK, also send contents of uri.

First looks for file called -, then .

FIXME: normal url syntax (NN should be interpreted as 0xNN) should be understood.

Parameters:

socket The socket to write to.

resp The reponse code.

uri The requested resource.

width The screen width given by the client.

height The screen height given by the client.

Returns:

1 on success, 0 on failure (ETP protocol errors do not count as failure).

4.10.2.7 char* response_description (int *response*)

Returns:

A string describing a response or NULL if not available.

4.10.2.8 int response_error (int *socket*, int *resp*)

Send an error response of type resp.

Parameters:

socket The socket to write to.

resp The response error code.

Returns:

1 on success, 0 otherwise.

4.10.2.9 ssize_t slow_send (int *socket*, const void * *msg*, size_t *len*, int *flags*)

Send 200 bytes of data, wait 0.3 s and repeat. This function is only used when ERICSSON_T300_COMPATIBILITY_MODE is defined.

Parameters:

socket The socket on which to send data.

msg The data to send.

len Size of data to send.

flags Flags to give to send(3).

Returns:

number of bytes sent, -1 if failed.

4.11 server.c File Reference

```
#include "common.h"
#include "error.h"
#include "server.h"
#include "net.h"
#include "commandline.h"
```

Functions

- int **main** (int argc, char ****argv**)
- RETSIGTYPE **signal_handler** (int signal)

Variables

- int **server_port** = 4711
- char * **server_root** = "."
- int **reverse_lookup** = 1
- int **run_as_daemon** = 1
- int **listen_socket**
- int **connection_socket**

4.11.1 Function Documentation

4.11.1.1 int main (int *argc*, char ** *argv*)

The Elbonia Server. Start listen for incoming connections and spawn child processes to take care of them.

4.11.1.2 RETSIGTYPE signal_handler (int *signal*)

Exit program on C-c. Close the listening socket first.

Parameters:

signal The signal being sent.

4.11.2 Variable Documentation

4.11.2.1 int connection_socket

4.11.2.2 int listen_socket

4.11.2.3 int reverse_lookup = 1

whether to do reverse lookups.

4.11.2.4 int run_as_daemon = 1

whether to run as daemon.

4.11.2.5 int server_port = 4711

port for listening for connections.

4.11.2.6 char* server_root = "."

server root directory.

4.12 server.h File Reference

Functions

- RETSIGTYPE **signal_handler** (int)

4.12.1 Function Documentation

4.12.1.1 RETSIGTYPE signal_handler (int *signal*)

Exit program on C-c. Close the listening socket first.

Parameters:

signal The signal being sent.

4.13 xmalloc.c File Reference

```
#include "common.h"
#include "error.h"
```

Functions

- void * **xmalloc** (size_t num)
- void * **xrealloc** (void *p, size_t num)
- void * **xcalloc** (size_t num, size_t size)
- char * **xstrdup** (const char *string)

4.13.1 Function Documentation

4.13.1.1 void* xcalloc (size_t *num*, size_t *size*)

Allocate and zero memory with sanity checking. Raises a fatal error if memory can't be allocated.

Parameters:

num Number of elements to allocate.

size Size of each element.

4.13.1.2 void* xmalloc (size_t *num*)

Allocate memory with sanity checking. Raises a fatal error if memory can't be allocated.

Parameters:

num Number of bytes to allocate.

Returns:

Pointer to allocated memory.

4.13.1.3 void* xrealloc (void * *p*, size_t *num*)

Reallocate memory with sanity checking. Raises a fatal error if memory can't be reallocated.

Parameters:

p Pointer to already allocated memory.

num Number bytes to allocate.

Returns:

Pointer to allocated memory.

4.13.1.4 char* xstrdup (const char * *string*)

Duplicate a string. Raises fatal error if memory for the string can't be allocated.

Parameters:
string String to duplicate.

Returns:
A copy of the string.

Index