ws

1.0

Generated by Doxygen 1.8.9.1

Sun Jan 25 2015 12:15:57

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 http_req Struct Reference

`#include <simple_http.h>`

Data Fields

- int fd
- char * request
- int req_len
- char * path
- char * resp_head
- char * response
- int resp_hd_len
- int resp_len

3.1.1 Detailed Description

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

3.1.2 Field Documentation

3.1.2.1 int http_req::fd

3.1.2.2 char* http_req::path

3.1.2.3 int http_req::req_len

3.1.2.4 char* http_req::request

3.1.2.5 int http_req::resp_hd_len

3.1.2.6 char* http_req::resp_head

3.1.2.7 int http_req::resp_len

3.1.2.8 char * http_req::response

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

- simple_http.h

3.2 list_t Struct Reference

list_t Linked list structure Holds pointers to head and tail

`#include <linkedlist.h>`

Data Fields

- node_t * head
- node_t * tail

3.2.1 Detailed Description

list_t Linked list structure Holds pointers to head and tail

3.2.2 Field Documentation

3.2.2.1 node_t* list_t::head

3.2.2.2 node_t* list_t::tail

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

- linkedlist.h

3.3 node_t Struct Reference

node_t Basic Node structure Holds pointer to next node and value

`#include <linkedlist.h>`

Data Fields

- node_t * next
- void * value

3.3.1 Detailed Description

node_t Basic Node structure Holds pointer to next node and value

3.3.2 Field Documentation

3.3.2.1 node_t* node_t::next

3.3.2.2 void* node_t::value

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

- linkedlist.h

3.4 pool_arg_t Struct Reference

pool_arg_t Args needed by the processor threads

`#include <threadpool.h>`

Data Fields

- void * args
- pool_t * pool
- pthread_t * curr_thread

3.4.1 Detailed Description

pool_arg_t Args needed by the processor threads

3.4.2 Field Documentation

3.4.2.1 void* pool_arg_t::args

3.4.2.2 pthread_t* pool_arg_t::curr_thread

3.4.2.3 pool_t* pool_arg_t::pool

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

- threadpool.h

3.5 pool_t Struct Reference

pool_t The thread pool and its metadata

`#include <threadpool.h>`

Data Fields

- list_t * list

- int current_size
- int max_size
- pthread_attr_t * attr
- pthread_mutex_t * count_mutex
- pthread_cond_t * count_cv

3.5.1 Detailed Description

pool_t The thread pool and its metadata

3.5.2 Field Documentation

3.5.2.1 pthread_attr_t* pool_t::attr

3.5.2.2 pthread_cond_t* pool_t::count_cv

3.5.2.3 pthread_mutex_t* pool_t::count_mutex

3.5.2.4 int pool_t::current_size

3.5.2.5 list_t* pool_t::list

3.5.2.6 int pool_t::max_size

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

- threadpool.h

Chapter 4

File Documentation

4.1 cas.h File Reference

4.2 content.c File Reference

```
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

Macros

- #define MAX_CONTENT_SZ (1024∗1024∗10)

Functions

- char ∗ error_resp (char ∗path, int ∗len)
- int sanity_check (char ∗path)
- char ∗ content_get (char ∗path, int ∗content_len)

4.2.1 Macro Definition Documentation

4.2.1.1 #define MAX_CONTENT_SZ (1024∗1024∗10)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.2.2 Function Documentation

4.2.2.1 char* content_get (char * *path*, int * *content_len*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.2.2.2 char* error_resp (char * *path*, int * *len*)

4.2.2.3 int sanity_check (char * *path*)

4.3 content.h File Reference

Functions

- char * content_get (char *path, int *content_len)

4.3.1 Function Documentation

4.3.1.1 char* content_get (char * *path*, int * *content_len*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.4 linkedlist.c File Reference

```
#import <linkedlist.h>
#import <stdlib.h>
#import <assert.h>
#import <stdio.h>
```

Functions

- node_t * create_node (void *value)

  create_node Allocates and intiializes a node with the provided value
- void push_node (node_t **head, void *value)

  push_node Pushes the value to the front of the lsit
- void * pop_node (node_t **head)

  pop_node Pops the node from the head
- node_t * remove_node (node_t **head, void *value)

  remove_node Removes a node from the list
- void init_list (list_t *list)

  init_list Initializes the list
- void * peek_list (list_t *list)

  peek_list Peeks at the head of the list and returns its value
- void * pop_list (list_t *list)

  pop_list Pops the head of the list and returns its value

- void push_list (list_t *list, void *value)

  push_list Pushes the node to the end of the list
- void remove_list (list_t *list, void *value)

  remove_list Removes the value from the list

4.4.1 Function Documentation

4.4.1.1 node_t* create_node (void * *value*)

create_node Allocates and intiializes a node with the provided value

Parameters

value	The value to use

Returns

A new node

4.4.1.2 void init_list (list_t * *list*)

init_list Initializes the list

Parameters

list	The list to init

4.4.1.3 void* peek_list (list_t * *list*)

peek_list Peeks at the head of the list and returns its value

Parameters

list	The list

Returns

The value of the head

4.4.1.4 void* pop_list (list_t * *list*)

pop_list Pops the head of the list and returns its value

Parameters

list	The list

Returns

The value of the head

4.4.1.5 void* pop_node (node_t ** *head*)

pop_node Pops the node from the head

Parameters

head	The head of the list

Returns

The value at that head

4.4.1.6 void push_list (list_t * *list*, void * *value*)

push_list Pushes the node to the end of the list

Parameters

list	The list to use
value	The value to store in the list

4.4.1.7 void push_node (node_t ** *head*, void * *value*)

push_node Pushes the value to the front of the lsit

Parameters

head	The pointer to the front of the list
value	The value of the node to use

4.4.1.8 void remove_list (list_t * *list*, void * *value*)

remove_list Removes the value from the list

Parameters

list	The list to use
value	The value to look for

4.4.1.9 node_t* remove_node (node_t ** *head*, void * *value*)

remove_node Removes a node from the list

Parameters

head	The head of the list
value	The value to search for

Returns

The detached node if it exists in the list

4.5 linkedlist.h File Reference

Data Structures

- struct node_t

  node_t Basic Node structure Holds pointer to next node and value
- struct list_t

  list_t Linked list structure Holds pointers to head and tail

Functions

- node_t * create_node (void *value)

  create_node Allocates and intiializes a node with the provided value
- void push_node (node_t **head, void *value)

  push_node Pushes the value to the front of the lsit
- void * pop_node (node_t **head)

  pop_node Pops the node from the head
- node_t * remove_node (node_t **head, void *value)

  remove_node Removes a node from the list
- void * peek_list (list_t *list)

  peek_list Peeks at the head of the list and returns its value
- void * pop_list (list_t *list)

  pop_list Pops the head of the list and returns its value
- void push_list (list_t *list, void *value)

  push_list Pushes the node to the end of the list
- void remove_list (list_t *list, void *value)

  remove_list Removes the value from the list
- void init_list (list_t *list)

  init_list Initializes the list

4.5.1 Function Documentation

4.5.1.1 node_t* create_node (void * *value*)

create_node Allocates and intiializes a node with the provided value

Parameters

value	The value to use

Returns

A new node

4.5.1.2 void init_list (list_t * *list*)

init_list Initializes the list

Parameters

list	The list to init

4.5.1.3 void* peek_list (list_t * *list*)

peek_list Peeks at the head of the list and returns its value

Parameters

list	The list

Returns

The value of the head

4.5.1.4 void* pop_list (list_t * *list*)

pop_list Pops the head of the list and returns its value

Parameters

list	The list

Returns

The value of the head

4.5.1.5 void* pop_node (node_t ** *head*)

pop_node Pops the node from the head

Parameters

head	The head of the list

Returns

The value at that head

4.5.1.6 void push_list (list_t * *list,* void * *value*)

push_list Pushes the node to the end of the list

Parameters

list	The list to use

value	The value to store in the list

4.5.1.7 void push_node (node_t ** *head*, void * *value*)

push_node Pushes the value to the front of the lsit

Parameters

head	The pointer to the front of the list
value	The value of the node to use

4.5.1.8 void remove_list (list_t * *list*, void * *value*)

remove_list Removes the value from the list

Parameters

list	The list to use
value	The value to look for

4.5.1.9 node_t* remove_node (node_t ** *head*, void * *value*)

remove_node Removes a node from the list

Parameters

head	The head of the list
value	The value to search for

Returns

The detached node if it exists in the list

4.6 main.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include <sys/wait.h>
#include <pthread.h>
#include <util.h>
#include <server.h>
#include <thread_per_request.h>
#include <thread_pool_request.h>
#include <cas.h>
```

Macros

- #define MAX_DATA_SZ 1024
- #define MAX_CONCURRENCY 4
- #define BUFFER_LENGTH 256

Enumerations

- enum server_type_t { SERVER_TYPE_ONE = 0, SERVER_TYPE_THREAD_PER_REQUEST, SERVER_TY↩ PE_THREAD_POOL_BOUND }

Functions

- void server_single_request (int accept_fd)
- void server_thread_per_req (int accept_fd)
- void server_thread_pool_bounded (int accept_fd)
- int main (int argc, char ∗argv[ ])

4.6.1 Macro Definition Documentation

4.6.1.1 #define BUFFER_LENGTH 256

4.6.1.2 #define MAX_CONCURRENCY 4

4.6.1.3 #define MAX_DATA_SZ 1024

 Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.6.2 Enumeration Type Documentation

4.6.2.1 enum server_type_t

Enumerator

SERVER_TYPE_ONE

SERVER_TYPE_THREAD_PER_REQUEST

SERVER_TYPE_THREAD_POOL_BOUND

4.6.3 Function Documentation

4.6.3.1 int main (int *argc,* char ∗ *argv[ ]*)

4.6.3.2 void server_single_request (int *accept_fd*)

4.6.3.3 void server_thread_per_req (int *accept_fd*)

4.6.3.4 void server_thread_pool_bounded (int *accept_fd*)

4.7 server.c File Reference

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <errno.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
```

Functions

- int server_create (short int port)
- int server_accept (int fd)

4.7.1 Function Documentation

4.7.1.1 int server_accept (int *fd*)

4.7.1.2 int server_create (short int *port*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.8 server.h File Reference

Functions

- int server_create (short int port)
- int server_accept (int fd)

4.8.1 Function Documentation

4.8.1.1 int server_accept (int *fd*)

4.8.1.2 int server_create (short int *port*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.9 simple_http.c File Reference

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

```
#include <assert.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <simple_http.h>
```

Macros

- #define MAX_DIGITS 128

Functions

- struct http_req * shttp_alloc_req (int fd, char *request)
- void shttp_free_req (struct http_req *r)
- int shttp_get_path (struct http_req *r)
- int shttp_alloc_response_head (struct http_req *r, char *data, int dlen)

4.9.1 Macro Definition Documentation

4.9.1.1 #define MAX_DIGITS 128

4.9.2 Function Documentation

4.9.2.1 struct http_req* shttp_alloc_req (int *fd,* char * *request*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.9.2.2 int shttp_alloc_response_head (struct http_req * *r,* char * *data,* int *dlen*)

4.9.2.3 void shttp_free_req (struct http_req * *r*)

4.9.2.4 int shttp_get_path (struct http_req * *r*)

4.10 simple_http.h File Reference

Data Structures

- struct http_req

Functions

- struct http_req * shttp_alloc_req (int fd, char *request)
- void shttp_free_req (struct http_req *r)
- int shttp_get_path (struct http_req *r)
- int shttp_alloc_response_head (struct http_req *r, char *resp, int rlen)

4.10.1 Function Documentation

4.10.1.1 struct http_req* shttp_alloc_req (int *fd*, char * *request*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.10.1.2 int shttp_alloc_response_head (struct http_req * *r*, char * *resp*, int *rlen*)

4.10.1.3 void shttp_free_req (struct http_req * *r*)

4.10.1.4 int shttp_get_path (struct http_req * *r*)

4.11 thread_per_request.c File Reference

```
#include <pthread.h>
#include <stdlib.h>
#include <server.h>
#include <util.h>
#include <assert.h>
#include <linkedlist.h>
#include <unistd.h>
#include <stdio.h>
```

Functions

- void * thread_process (void *fd)

  thread_process Function to execute on the threads; services the client's request
- void process_threads_per_request (int max_concurrency, int accept_fd)

  process_threads_per_request Processes the request using a thread per request

4.11.1 Function Documentation

4.11.1.1 void process_threads_per_request (int *max_concurrency*, int *accept_fd*)

process_threads_per_request Processes the request using a thread per request

Parameters

max_concurrency	The maximum number of concurrent threads
accept_fd	The accepted file descriptor

4.11.1.2 void* thread_process (void * *fd*)

thread_process Function to execute on the threads; services the client's request

Parameters

fd	The file descriptor of the opened request

Returns

0

4.12 thread_per_request.h File Reference

Functions

- void process_threads_per_request (int max_concurrency, int accept_fd)

  process_threads_per_request Processes the request using a thread per request

4.12.1 Function Documentation

4.12.1.1 void process_threads_per_request (int *max_concurrency*, int *accept_fd*)

process_threads_per_request Processes the request using a thread per request

Parameters

max_concurrency	The maximum number of concurrent threads
accept_fd	The accepted file descriptor

4.13 thread_pool_request.c File Reference

```
#include <stdlib.h>
#include <threadpool.h>
#include <server.h>
#include <util.h>
#include <stdio.h>
#include <assert.h>
```

Functions

- void * thread_process_in_pool (void *args)

  thread_process_in_pool Thread function; services the client request

- int server_process (void *args)

  server_process The server function pointer; accepts the request

- void process_request_thread_pool (int max_size, int accept_fd)

  process_request_thread_pool Creates and runs a thread pool

4.13.1 Function Documentation

4.13.1.1 void process_request_thread_pool (int *max_size,* int *accept_fd*)

process_request_thread_pool Creates and runs a thread pool

Parameters

max_size	The max size of the thread pool
accept_fd	The accepted file descriptor of the listener

4.13.1.2 int server_process (void * *args*)

server_process The server function pointer; accepts the request

Parameters

args	The file descriptor of the port being listened

Returns

A new file descriptor for the accepted connection

4.13.1.3 void* thread_process_in_pool (void * *args*)

thread_process_in_pool Thread function; services the client request

Parameters

args	The pool_args

Returns

0

4.14 thread_pool_request.h File Reference

Functions

- void process_request_thread_pool (int max_size, int accept_fd)

  process_request_thread_pool Creates and runs a thread pool

4.14.1 Function Documentation

4.14.1.1 void process_request_thread_pool (int *max_size,* int *accept_fd*)

process_request_thread_pool Creates and runs a thread pool

Parameters

max_size	The max size of the thread pool
accept_fd	The accepted file descriptor of the listener

4.15 threadpool.c File Reference

```
#include <pthread.h>
#include <threadpool.h>
#include <linkedlist.h>
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
```

Functions

- pool_t * init_pool (int max_size)

  init_pool Allocates and intializes the thread pool
- void free_pool (pool_t *pool)

  free_pool Frees the pool and its resources
- void add_to_pool (pool_t *pool, void *(*server_routine)(void *), void *(*start_routine)(void *), void *args)

  add_to_pool Adds a connection to be processed in the pool
- void remove_from_pool (pool_t *pool, pthread_t *thread)

  remove_from_pool Removes a task from the pool

4.15.1 Function Documentation

4.15.1.1 void add_to_pool (pool_t * *pool*, void *(*)(void *) *server_routine*, void *(*)(void *) *start_routine*, void * *args*)

add_to_pool Adds a connection to be processed in the pool

Parameters

pool	The thread pool
server_routine	The function pointer to the server routine
start_routine	The function pointer to the thread routine
args	The args for the server routine

4.15.1.2 void free_pool (pool_t * *pool*)

free_pool Frees the pool and its resources

Parameters

pool	The thread pool to be freed

4.15.1.3 pool_t* init_pool (int *max_size*)

init_pool Allocates and intializes the thread pool

Parameters

max_size	The max size of the pool

Returns

An initialized thread pool

4.15.1.4 void remove_from_pool (pool_t * *pool*, pthread_t * *thread*)

remove_from_pool Removes a task from the pool

Parameters

pool	The pool
thread	The task as represented by the thread id

4.16 threadpool.h File Reference

```
#import <linkedlist.h>
#import <pthread.h>
```

Data Structures

- struct pool_t

  pool_t The thread pool and its metadata
- struct pool_arg_t

  pool_arg_t Args needed by the processor threads

Functions

- pool_t * init_pool (int max_size)

  init_pool Allocates and intializes the thread pool
- void free_pool (pool_t *pool)

  free_pool Frees the pool and its resources
- void add_to_pool (pool_t *pool, void *(*server_routine)(void *), void *(*start_routine)(void *), void *args)

  add_to_pool Adds a connection to be processed in the pool
- void remove_from_pool (pool_t *pool, pthread_t *thread)

  remove_from_pool Removes a task from the pool

4.16.1 Function Documentation

4.16.1.1 void add_to_pool (pool_t * *pool*, void *(*)(void *) *server_routine*, void *(*)(void *) *start_routine*, void * *args*)

add_to_pool Adds a connection to be processed in the pool

Parameters

pool	The thread pool
server_routine	The function pointer to the server routine
start_routine	The function pointer to the thread routine
args	The args for the server routine

4.16.1.2 void free_pool (pool_t * *pool*)

free_pool Frees the pool and its resources

Parameters

pool	The thread pool to be freed

4.16.1.3 pool_t* init_pool (int *max_size*)

init_pool Allocates and intializes the thread pool

Parameters

max_size	The max size of the pool

Returns

An initialized thread pool

4.16.1.4 void remove_from_pool (pool_t * *pool*, pthread_t * *thread*)

remove_from_pool Removes a task from the pool

Parameters

pool	The pool
thread	The task as represented by the thread id

4.17 util.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include <server.h>
#include <simple_http.h>
#include <content.h>
```

Macros

- #define MAX_REQ_SZ 1024

Functions

- struct http_req * newfd_create_req (int new_fd)
- void respond_and_free_req (struct http_req *r, char *response, int len)
- void client_process (int fd)

4.17.1 Macro Definition Documentation

4.17.1.1 #define MAX_REQ_SZ 1024

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.17.2 Function Documentation

4.17.2.1 void client_process (int *fd*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.17.2.2 struct http_req* newfd_create_req (int *new_fd*)

4.17.2.3 void respond_and_free_req (struct http_req * *r,* char * *response,* int *len*)

4.18 util.h File Reference

Functions

- void client_process (int fd)

4.18.1 Function Documentation

4.18.1.1 void client_process (int *fd*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

Index