

WS
1.0

Generated by Doxygen 1.8.9.1

Sun Jan 25 2015 12:15:57

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	http_req Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	fd	5
3.1.2.2	path	5
3.1.2.3	req_len	5
3.1.2.4	request	5
3.1.2.5	resp_hd_len	5
3.1.2.6	resp_head	5
3.1.2.7	resp_len	6
3.1.2.8	response	6
3.2	list_t Struct Reference	6
3.2.1	Detailed Description	6
3.2.2	Field Documentation	6
3.2.2.1	head	6
3.2.2.2	tail	6
3.3	node_t Struct Reference	6
3.3.1	Detailed Description	6
3.3.2	Field Documentation	7
3.3.2.1	next	7
3.3.2.2	value	7
3.4	pool_arg_t Struct Reference	7

3.4.1	Detailed Description	7
3.4.2	Field Documentation	7
3.4.2.1	args	7
3.4.2.2	curr_thread	7
3.4.2.3	pool	7
3.5	pool_t Struct Reference	7
3.5.1	Detailed Description	8
3.5.2	Field Documentation	8
3.5.2.1	attr	8
3.5.2.2	count_cv	8
3.5.2.3	count_mutex	8
3.5.2.4	current_size	8
3.5.2.5	list	8
3.5.2.6	max_size	8
4	File Documentation	9
4.1	cas.h File Reference	9
4.2	content.c File Reference	9
4.2.1	Macro Definition Documentation	9
4.2.1.1	MAX_CONTENT_SZ	9
4.2.2	Function Documentation	9
4.2.2.1	content_get	10
4.2.2.2	error_resp	10
4.2.2.3	sanity_check	10
4.3	content.h File Reference	10
4.3.1	Function Documentation	10
4.3.1.1	content_get	10
4.4	linkedlist.c File Reference	10
4.4.1	Function Documentation	11
4.4.1.1	create_node	11
4.4.1.2	init_list	11
4.4.1.3	peek_list	11
4.4.1.4	pop_list	11
4.4.1.5	pop_node	12
4.4.1.6	push_list	13
4.4.1.7	push_node	13
4.4.1.8	remove_list	13

4.4.1.9	remove_node	13
4.5	linkedlist.h File Reference	13
4.5.1	Function Documentation	14
4.5.1.1	create_node	14
4.5.1.2	init_list	14
4.5.1.3	peek_list	15
4.5.1.4	pop_list	15
4.5.1.5	pop_node	15
4.5.1.6	push_list	15
4.5.1.7	push_node	16
4.5.1.8	remove_list	16
4.5.1.9	remove_node	16
4.6	main.c File Reference	16
4.6.1	Macro Definition Documentation	17
4.6.1.1	BUFFER_LENGTH	17
4.6.1.2	MAX_CONCURRENCY	17
4.6.1.3	MAX_DATA_SZ	17
4.6.2	Enumeration Type Documentation	17
4.6.2.1	server_type_t	17
4.6.3	Function Documentation	17
4.6.3.1	main	17
4.6.3.2	server_single_request	17
4.6.3.3	server_thread_per_req	17
4.6.3.4	server_thread_pool_bounded	17
4.7	server.c File Reference	18
4.7.1	Function Documentation	18
4.7.1.1	server_accept	18
4.7.1.2	server_create	18
4.8	server.h File Reference	18
4.8.1	Function Documentation	18
4.8.1.1	server_accept	18
4.8.1.2	server_create	18
4.9	simple_http.c File Reference	18
4.9.1	Macro Definition Documentation	19
4.9.1.1	MAX_DIGITS	19
4.9.2	Function Documentation	19
4.9.2.1	shttp_alloc_req	19

4.9.2.2	shhttp_alloc_response_head	19
4.9.2.3	shhttp_free_req	19
4.9.2.4	shhttp_get_path	19
4.10	simple_http.h File Reference	19
4.10.1	Function Documentation	20
4.10.1.1	shhttp_alloc_req	20
4.10.1.2	shhttp_alloc_response_head	20
4.10.1.3	shhttp_free_req	20
4.10.1.4	shhttp_get_path	20
4.11	thread_per_request.c File Reference	20
4.11.1	Function Documentation	20
4.11.1.1	process_threads_per_request	20
4.11.1.2	thread_process	20
4.12	thread_per_request.h File Reference	21
4.12.1	Function Documentation	21
4.12.1.1	process_threads_per_request	21
4.13	thread_pool_request.c File Reference	21
4.13.1	Function Documentation	22
4.13.1.1	process_request_thread_pool	22
4.13.1.2	server_process	23
4.13.1.3	thread_process_in_pool	23
4.14	thread_pool_request.h File Reference	23
4.14.1	Function Documentation	23
4.14.1.1	process_request_thread_pool	23
4.15	threadpool.c File Reference	24
4.15.1	Function Documentation	24
4.15.1.1	add_to_pool	24
4.15.1.2	free_pool	24
4.15.1.3	init_pool	24
4.15.1.4	remove_from_pool	25
4.16	threadpool.h File Reference	25
4.16.1	Function Documentation	25
4.16.1.1	add_to_pool	25
4.16.1.2	free_pool	26
4.16.1.3	init_pool	26
4.16.1.4	remove_from_pool	26
4.17	util.c File Reference	26

4.17.1	Macro Definition Documentation	27
4.17.1.1	MAX_REQ_SZ	27
4.17.2	Function Documentation	27
4.17.2.1	client_process	27
4.17.2.2	newfd_create_req	27
4.17.2.3	respond_and_free_req	27
4.18	util.h File Reference	27
4.18.1	Function Documentation	27
4.18.1.1	client_process	27
Index		29

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

http_req	5
list_t	
	List_t Linked list structure Holds pointers to head and tail	6
node_t	
	Node_t Basic Node structure Holds pointer to next node and value	6
pool_arg_t	
	Pool_arg_t Args needed by the processor threads	7
pool_t	
	Pool_t The thread pool and its metadata	7

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

cas.h	9
content.c	9
content.h	10
linkedList.c	10
linkedList.h	13
main.c	16
server.c	18
server.h	18
simple_http.c	18
simple_http.h	19
thread_per_request.c	20
thread_per_request.h	21
thread_pool_request.c	21
thread_pool_request.h	23
threadpool.c	24
threadpool.h	25
util.c	26
util.h	27

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 initializes a node with the provided value
- `void push_node (node_t **head, void *value)`
push_node Pushes the value to the front of the list
- `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 initializes a node with the provided value

Parameters

<i>value</i>	The value to use
--------------	------------------

Returns

A new node

4.4.1.2 `void init_list (list_t * list)`

`init_list` Initializes the list

Parameters

<i>list</i>	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

<i>list</i>	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

<i>list</i>	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

<i>head</i>	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

<i>list</i>	The list to use
<i>value</i>	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 list

Parameters

<i>head</i>	The pointer to the front of the list
<i>value</i>	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

<i>list</i>	The list to use
<i>value</i>	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

<i>head</i>	The head of the list
<i>value</i>	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 list
- `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

<i>value</i>	The value to use
--------------	------------------

Returns

A new node

4.5.1.2 `void init_list (list_t * list)`

`init_list` Initializes the list

Parameters

<i>list</i>	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

<i>list</i>	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

<i>list</i>	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

<i>head</i>	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

<i>list</i>	The list to use
-------------	-----------------

<i>value</i>	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 list

Parameters

<i>head</i>	The pointer to the front of the list
<i>value</i>	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

<i>list</i>	The list to use
<i>value</i>	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

<i>head</i>	The head of the list
<i>value</i>	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_TYPE_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

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

4.6.2 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](#) * [shhttp_alloc_req](#) (int fd, char *request)
- void [shhttp_free_req](#) (struct [http_req](#) *r)
- int [shhttp_get_path](#) (struct [http_req](#) *r)
- int [shhttp_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](#)* [shhttp_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 [shhttp_alloc_response_head](#) (struct [http_req](#) * *r*, char * *data*, int *dlen*)

4.9.2.3 void [shhttp_free_req](#) (struct [http_req](#) * *r*)

4.9.2.4 int [shhttp_get_path](#) (struct [http_req](#) * *r*)

4.10 simple_http.h File Reference

Data Structures

- struct [http_req](#)

Functions

- struct [http_req](#) * [shhttp_alloc_req](#) (int fd, char *request)
- void [shhttp_free_req](#) (struct [http_req](#) *r)
- int [shhttp_get_path](#) (struct [http_req](#) *r)
- int [shhttp_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

<i>max_concurrency</i>	The maximum number of concurrent threads
<i>accept_fd</i>	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

<i>fd</i>	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

<i>max_↔ concurrency</i>	The maximum number of concurrent threads
<i>accept_fd</i>	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

<i>max_size</i>	The max size of the thread pool
<i>accept_fd</i>	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

<i>args</i>	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

<i>args</i>	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

<i>max_size</i>	The max size of the thread pool
<i>accept_fd</i>	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 initializes 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 (*)(void *) server_routine, void (*)(void *) start_routine, 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

<i>pool</i>	The thread pool
<i>server_routine</i>	The function pointer to the server routine
<i>start_routine</i>	The function pointer to the thread routine
<i>args</i>	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

<i>pool</i>	The thread pool to be freed
-------------	-----------------------------

4.15.1.3 `pool_t* init_pool (int max_size)`

`init_pool` Allocates and initializes the thread pool

Parameters

<i>max_size</i>	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

<i>pool</i>	The pool
<i>thread</i>	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 initializes 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 (*)(void *) *server_routine*, void (*)(void *) *start_routine*, 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

<i>pool</i>	The thread pool
<i>server_routine</i>	The function pointer to the server routine
<i>start_routine</i>	The function pointer to the thread routine
<i>args</i>	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

<i>pool</i>	The thread pool to be freed
-------------	-----------------------------

4.16.1.3 pool_t* init_pool (int max_size)

init_pool Allocates and initializes the thread pool

Parameters

<i>max_size</i>	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

<i>pool</i>	The pool
<i>thread</i>	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

- add_to_pool
 - threadpool.c, [24](#)
 - threadpool.h, [25](#)
- args
 - pool_arg, [7](#)
- attr
 - pool, [8](#)
- BUFFER_LENGTH
 - main.c, [17](#)
- cas.h, [9](#)
- client_process
 - util.c, [27](#)
 - util.h, [27](#)
- content.c, [9](#)
 - content_get, [9](#)
 - error_resp, [10](#)
 - MAX_CONTENT_SZ, [9](#)
 - sanity_check, [10](#)
- content.h, [10](#)
 - content_get, [10](#)
- content_get
 - content.c, [9](#)
 - content.h, [10](#)
- count_cv
 - pool, [8](#)
- count_mutex
 - pool, [8](#)
- create_node
 - linkedlist.c, [11](#)
 - linkedlist.h, [14](#)
- curr_thread
 - pool_arg, [7](#)
- current_size
 - pool, [8](#)
- error_resp
 - content.c, [10](#)
- fd
 - http_req, [5](#)
- free_pool
 - threadpool.c, [24](#)
 - threadpool.h, [26](#)
- head
 - list, [6](#)
- http_req, [5](#)
 - fd, [5](#)
 - path, [5](#)
 - req_len, [5](#)
 - request, [5](#)
 - resp_hd_len, [5](#)
 - resp_head, [5](#)
 - resp_len, [5](#)
 - response, [6](#)
- init_list
 - linkedlist.c, [11](#)
 - linkedlist.h, [14](#)
- init_pool
 - threadpool.c, [24](#)
 - threadpool.h, [26](#)
- linkedlist.c, [10](#)
 - create_node, [11](#)
 - init_list, [11](#)
 - peek_list, [11](#)
 - pop_list, [11](#)
 - pop_node, [11](#)
 - push_list, [13](#)
 - push_node, [13](#)
 - remove_list, [13](#)
 - remove_node, [13](#)
- linkedlist.h, [13](#)
 - create_node, [14](#)
 - init_list, [14](#)
 - peek_list, [15](#)
 - pop_list, [15](#)
 - pop_node, [15](#)
 - push_list, [15](#)
 - push_node, [16](#)
 - remove_list, [16](#)
 - remove_node, [16](#)
- list
 - head, [6](#)
 - pool, [8](#)
 - tail, [6](#)
- list_t, [6](#)
- MAX_CONCURRENCY
 - main.c, [17](#)

- MAX_CONTENT_SZ
 - content.c, 9
- MAX_DATA_SZ
 - main.c, 17
- MAX_DIGITS
 - simple_http.c, 19
- MAX_REQ_SZ
 - util.c, 27
- main
 - main.c, 17
- main.c, 16
 - BUFFER_LENGTH, 17
 - MAX_CONCURRENCY, 17
 - MAX_DATA_SZ, 17
 - main, 17
 - SERVER_TYPE_ONE, 17
 - SERVER_TYPE_THREAD_PER_REQUEST, 17
 - SERVER_TYPE_THREAD_POOL_BOUND, 17
 - server_single_request, 17
 - server_thread_per_req, 17
 - server_thread_pool_bounded, 17
 - server_type_t, 17
- max_size
 - pool, 8
- newfd_create_req
 - util.c, 27
- next
 - node, 7
- node
 - next, 7
 - value, 7
- node_t, 6
- path
 - http_req, 5
- peek_list
 - linkedlist.c, 11
 - linkedlist.h, 15
- pool
 - attr, 8
 - count_cv, 8
 - count_mutex, 8
 - current_size, 8
 - list, 8
 - max_size, 8
 - pool_arg, 7
- pool_arg
 - args, 7
 - curr_thread, 7
 - pool, 7
- pool_arg_t, 7
- pool_t, 7
- pop_list
 - linkedlist.c, 11
 - linkedlist.h, 15
- pop_node
 - linkedlist.c, 11
 - linkedlist.h, 15
- process_request_thread_pool
 - thread_pool_request.c, 22
 - thread_pool_request.h, 23
- process_threads_per_request
 - thread_per_request.c, 20
 - thread_per_request.h, 21
- push_list
 - linkedlist.c, 13
 - linkedlist.h, 15
- push_node
 - linkedlist.c, 13
 - linkedlist.h, 16
- remove_from_pool
 - threadpool.c, 25
 - threadpool.h, 26
- remove_list
 - linkedlist.c, 13
 - linkedlist.h, 16
- remove_node
 - linkedlist.c, 13
 - linkedlist.h, 16
- req_len
 - http_req, 5
- request
 - http_req, 5
- resp_hd_len
 - http_req, 5
- resp_head
 - http_req, 5
- resp_len
 - http_req, 5
- respond_and_free_req
 - util.c, 27
- response
 - http_req, 6
- SERVER_TYPE_ONE
 - main.c, 17
- SERVER_TYPE_THREAD_PER_REQUEST
 - main.c, 17
- SERVER_TYPE_THREAD_POOL_BOUND
 - main.c, 17
- sanity_check
 - content.c, 10
- server.c, 18
 - server_accept, 18
 - server_create, 18
- server.h, 18
 - server_accept, 18
 - server_create, 18

- server_accept
 - server.c, [18](#)
 - server.h, [18](#)
- server_create
 - server.c, [18](#)
 - server.h, [18](#)
- server_process
 - thread_pool_request.c, [23](#)
- server_single_request
 - main.c, [17](#)
- server_thread_per_req
 - main.c, [17](#)
- server_thread_pool_bounded
 - main.c, [17](#)
- server_type_t
 - main.c, [17](#)
- shttp_alloc_req
 - simple_http.c, [19](#)
 - simple_http.h, [20](#)
- shttp_alloc_response_head
 - simple_http.c, [19](#)
 - simple_http.h, [20](#)
- shttp_free_req
 - simple_http.c, [19](#)
 - simple_http.h, [20](#)
- shttp_get_path
 - simple_http.c, [19](#)
 - simple_http.h, [20](#)
- simple_http.c, [18](#)
 - MAX_DIGITS, [19](#)
 - shttp_alloc_req, [19](#)
 - shttp_alloc_response_head, [19](#)
 - shttp_free_req, [19](#)
 - shttp_get_path, [19](#)
- simple_http.h, [19](#)
 - shttp_alloc_req, [20](#)
 - shttp_alloc_response_head, [20](#)
 - shttp_free_req, [20](#)
 - shttp_get_path, [20](#)
- tail
 - list, [6](#)
- thread_per_request.c, [20](#)
 - process_threads_per_request, [20](#)
 - thread_process, [20](#)
- thread_per_request.h, [21](#)
 - process_threads_per_request, [21](#)
- thread_pool_request.c, [21](#)
 - process_request_thread_pool, [22](#)
 - server_process, [23](#)
 - thread_process_in_pool, [23](#)
- thread_pool_request.h, [23](#)
 - process_request_thread_pool, [23](#)
- thread_process
 - thread_per_request.c, [20](#)
- thread_process_in_pool
 - thread_pool_request.c, [23](#)
- threadpool.c, [24](#)
 - add_to_pool, [24](#)
 - free_pool, [24](#)
 - init_pool, [24](#)
 - remove_from_pool, [25](#)
- threadpool.h, [25](#)
 - add_to_pool, [25](#)
 - free_pool, [26](#)
 - init_pool, [26](#)
 - remove_from_pool, [26](#)
- util.c, [26](#)
 - client_process, [27](#)
 - MAX_REQ_SZ, [27](#)
 - newfd_create_req, [27](#)
 - respond_and_free_req, [27](#)
- util.h, [27](#)
 - client_process, [27](#)
- value
 - node, [7](#)