

Zappy

Generated by Doxygen 1.8.14

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	server/circular_buffer/add_to_buffer.c File Reference	3
2.1.1	Function Documentation	3
2.1.1.1	add_to_buffer()	3
2.1.1.2	can_fit_in_buffer()	4
2.1.1.3	get_len_left()	4
2.2	server/circular_buffer/has_command.c File Reference	4
2.2.1	Function Documentation	5
2.2.1.1	has_command()	5
2.3	server/circular_buffer/init_buffer.c File Reference	5
2.3.1	Function Documentation	5
2.3.1.1	init_c_buffer()	5
2.4	server/circular_buffer/read_from_buffer.c File Reference	6
2.4.1	Function Documentation	6
2.4.1.1	is_buf_readable()	6
2.4.1.2	read_from_buffer()	6
2.5	server/clock/check_for_timeouts.c File Reference	7
2.5.1	Function Documentation	7
2.5.1.1	check_for_timeouts()	7
2.6	server/clock/clock_tick.c File Reference	7
2.6.1	Function Documentation	8

2.6.1.1	<code>clock_tick()</code>	8
2.6.1.2	<code>clock_tick_clients()</code>	8
2.6.1.3	<code>clock_tick_player_ttl()</code>	9
2.6.1.4	<code>clock_tick_players()</code>	9
2.7	<code>server/game/commands/broadcast.c</code> File Reference	9
2.7.1	Function Documentation	10
2.7.1.1	<code>cmd_broadcast()</code>	10
2.7.1.2	<code>convert_angle_to_num()</code>	10
2.7.1.3	<code>get_angle()</code>	11
2.7.1.4	<code>get_origin()</code>	11
2.7.1.5	<code>get_traj()</code>	12
2.8	<code>server/game/commands/command_list.c</code> File Reference	12
2.8.1	Variable Documentation	12
2.8.1.1	<code>command_list</code>	12
2.9	<code>server/game/commands/connect_nbr.c</code> File Reference	12
2.9.1	Function Documentation	13
2.9.1.1	<code>cmd_connect_nbr()</code>	13
2.10	<code>server/game/commands/eject.c</code> File Reference	13
2.10.1	Function Documentation	13
2.10.1.1	<code>cmd_eject()</code>	14
2.10.1.2	<code>dir_to_str()</code>	14
2.10.1.3	<code>init_eject_msg()</code>	14
2.11	<code>server/game/commands/fork.c</code> File Reference	14
2.11.1	Function Documentation	15
2.11.1.1	<code>cmd_fork()</code>	15
2.12	<code>server/game/commands/incantation.c</code> File Reference	15
2.12.1	Function Documentation	16
2.12.1.1	<code>check_if_ritual_req_met()</code>	16
2.12.1.2	<code>cmd_incantation()</code>	16
2.12.1.3	<code>send_ko_to_ritual_players()</code>	17

2.12.1.4	upgrade_ritual_players()	17
2.12.2	Variable Documentation	17
2.12.2.1	ritual_data	17
2.13	server/game/commands/inventory.c File Reference	18
2.13.1	Function Documentation	18
2.13.1.1	cmd_inventory()	18
2.14	server/game/commands/look.c File Reference	18
2.14.1	Function Documentation	19
2.14.1.1	cmd_look()	19
2.14.1.2	fill_resp_look()	19
2.14.1.3	fill_resp_look_row()	20
2.15	server/game/commands/look_tile.c File Reference	20
2.15.1	Function Documentation	20
2.15.1.1	look_print()	20
2.15.1.2	look_print_player()	20
2.15.1.3	look_tile()	21
2.16	server/game/commands/move.c File Reference	21
2.16.1	Function Documentation	22
2.16.1.1	cmd_forward()	22
2.16.1.2	cmd_left()	22
2.16.1.3	cmd_right()	23
2.17	server/game/commands/take_set.c File Reference	23
2.17.1	Function Documentation	23
2.17.1.1	cmd_set()	23
2.17.1.2	cmd_take()	24
2.18	server/game/inventory/check_ritual_resources.c File Reference	24
2.18.1	Function Documentation	25
2.18.1.1	check_ritual_resources()	25
2.19	server/game/inventory/get_resource.c File Reference	25
2.19.1	Function Documentation	25

2.19.1.1	get_random_resource()	25
2.19.1.2	get_resource_from_str()	26
2.20	server/game/inventory/transfer_item.c File Reference	26
2.20.1	Function Documentation	26
2.20.1.1	add_resource_to_inventory()	26
2.20.1.2	del_resource_from_inventory()	27
2.20.1.3	transfer_item()	27
2.21	server/game/log/hatch_log.c File Reference	28
2.21.1	Function Documentation	28
2.21.1.1	hatch_log()	28
2.22	server/game/log/incantation_log.c File Reference	28
2.22.1	Function Documentation	29
2.22.1.1	incantation_end_log()	29
2.22.1.2	incantation_start_log()	29
2.23	server/game/log/kill_log.c File Reference	29
2.23.1	Function Documentation	30
2.23.1.1	kill_log()	30
2.24	server/game/log/move_log.c File Reference	30
2.24.1	Function Documentation	30
2.24.1.1	dir_log()	31
2.24.1.2	forward_log()	31
2.25	server/game/log/spawn_log.c File Reference	31
2.25.1	Function Documentation	31
2.25.1.1	spawn_player_log()	31
2.25.1.2	spawn_resource_log()	32
2.26	server/game/log/take_log.c File Reference	32
2.26.1	Function Documentation	33
2.26.1.1	take_log()	33
2.26.1.2	take_log_inventory()	33
2.27	server/game/map/add_player_to_tile.c File Reference	33

2.27.1	Function Documentation	34
2.27.1.1	append_to_tile_player_list()	34
2.28	server/game/map/get_dir.c File Reference	34
2.28.1	Function Documentation	34
2.28.1.1	get_dir_to_left()	34
2.28.1.2	get_dir_to_right()	35
2.29	server/game/map/get_nbr_players.c File Reference	35
2.29.1	Function Documentation	35
2.29.1.1	get_nbr_players_of_lvl_on_tile()	36
2.29.1.2	get_nbr_players_on_tile()	36
2.30	server/game/team/get_nbr_players.c File Reference	36
2.30.1	Function Documentation	37
2.30.1.1	get_team_nbr_players()	37
2.30.1.2	get_team_nbr_players_free()	37
2.30.1.3	get_team_nbr_players_not_free()	37
2.31	server/game/map/get_player.c File Reference	38
2.31.1	Function Documentation	38
2.31.1.1	get_last_tile_player()	38
2.32	server/game/team/get_player.c File Reference	38
2.32.1	Function Documentation	39
2.32.1.1	get_last_team_player()	39
2.33	server/game/map/get_tile.c File Reference	39
2.33.1	Function Documentation	39
2.33.1.1	get_random_tile()	39
2.33.1.2	get_tile()	40
2.33.1.3	get_tile_in_dir()	40
2.34	server/game/map/move_player_to_tile.c File Reference	41
2.34.1	Function Documentation	41
2.34.1.1	move_player_in_dir()	41
2.34.1.2	move_player_to_tile()	41

2.35	server/game/map/remove_player_from_tile.c File Reference	43
2.35.1	Function Documentation	43
2.35.1.1	remove_player_from_tile()	43
2.36	server/game/map/remove_resources_from_tile.c File Reference	44
2.36.1	Function Documentation	44
2.36.1.1	remove_resources_from_tile()	44
2.37	server/game/map/spawn_food.c File Reference	44
2.37.1	Function Documentation	45
2.37.1.1	init_random_pos()	45
2.37.1.2	spawn_resource()	45
2.37.1.3	spawn_resource_at_pos()	45
2.38	server/game/player/add.c File Reference	46
2.38.1	Function Documentation	46
2.38.1.1	create_player()	46
2.38.1.2	init_player_tile()	47
2.38.1.3	make_new_player()	47
2.39	server/network/client/add.c File Reference	48
2.39.1	Function Documentation	48
2.39.1.1	append_to_client_list()	48
2.40	server/game/player/append.c File Reference	48
2.40.1	Function Documentation	49
2.40.1.1	append_to_global_player_list()	49
2.41	server/game/player/check.c File Reference	49
2.41.1	Function Documentation	49
2.41.1.1	has_pending_commands()	49
2.42	server/game/player/delete.c File Reference	50
2.42.1	Function Documentation	50
2.42.1.1	delete_player()	50
2.42.1.2	free_all_players()	51
2.42.1.3	free_player()	51

2.42.1.4	remove_player()	51
2.42.1.5	remove_player_from_global_player_list()	52
2.43	server/network/client/delete.c File Reference	52
2.43.1	Function Documentation	52
2.43.1.1	delete_client()	52
2.43.1.2	disconnect_client()	53
2.43.1.3	free_all_clients()	53
2.43.1.4	free_client()	53
2.44	server/game/player/get.c File Reference	54
2.44.1	Function Documentation	54
2.44.1.1	get_last_global_player()	54
2.44.1.2	get_nbr_players_global()	54
2.45	server/network/client/get.c File Reference	55
2.45.1	Function Documentation	55
2.45.1.1	get_client_by_fd()	55
2.45.1.2	get_last_client()	55
2.46	server/game/team/add_player.c File Reference	56
2.46.1	Function Documentation	56
2.46.1.1	append_to_team_player_list()	56
2.47	server/game/team/get_team.c File Reference	57
2.47.1	Function Documentation	57
2.47.1.1	get_team()	57
2.48	server/game/team/remove_player.c File Reference	57
2.48.1	Function Documentation	58
2.48.1.1	remove_player_from_team()	58
2.49	server/game/update/check_if_game_won.c File Reference	58
2.49.1	Function Documentation	58
2.49.1.1	check_if_game_won()	58
2.50	server/game/update/execute_player_commands.c File Reference	59
2.50.1	Function Documentation	59

2.50.1.1	<code>execute_player_command()</code>	59
2.50.1.2	<code>fill_response_to_client()</code>	59
2.50.1.3	<code>strcmp_until_space()</code>	60
2.51	<code>server/game/update/set_player_command.c</code> File Reference	60
2.51.1	Function Documentation	60
2.51.1.1	<code>send_ritual_start_msg_to_players()</code>	60
2.51.1.2	<code>set_command()</code>	61
2.51.1.3	<code>set_incantation_command()</code>	61
2.51.1.4	<code>set_player_command()</code>	61
2.52	<code>server/game/update/update_game.c</code> File Reference	62
2.52.1	Function Documentation	62
2.52.1.1	<code>handle_player_command()</code>	62
2.52.1.2	<code>handle_players()</code>	62
2.52.1.3	<code>should_game_reset()</code>	63
2.52.1.4	<code>update_game()</code>	63
2.53	<code>server/get_args/get_args.c</code> File Reference	63
2.53.1	Function Documentation	63
2.53.1.1	<code>check_arg_holder()</code>	64
2.53.1.2	<code>free_arg_holder()</code>	64
2.53.1.3	<code>get_args()</code>	64
2.53.1.4	<code>get_args_loop()</code>	65
2.54	<code>server/get_args/options_func.c</code> File Reference	65
2.54.1	Function Documentation	65
2.54.1.1	<code>handle_opt_c()</code>	65
2.54.1.2	<code>handle_opt_f()</code>	66
2.54.1.3	<code>handle_opt_p()</code>	66
2.54.1.4	<code>handle_opt_x()</code>	66
2.54.1.5	<code>handle_opt_y()</code>	66
2.55	<code>server/get_args/options_names_func.c</code> File Reference	66
2.55.1	Function Documentation	67

2.55.1.1	handle_opt_n()	67
2.55.1.2	is_arg_option()	67
2.56	server/init/game/game.c File Reference	67
2.56.1	Function Documentation	67
2.56.1.1	init_game()	67
2.57	server/init/game/link_tiles.c File Reference	68
2.57.1	Function Documentation	68
2.57.1.1	link_tile()	68
2.57.1.2	link_tiles()	69
2.58	server/init/game/teams.c File Reference	69
2.58.1	Function Documentation	69
2.58.1.1	free_team_array()	69
2.58.1.2	init_team()	70
2.58.1.3	init_teams()	70
2.58.1.4	print_teams()	71
2.59	server/init/game/world.c File Reference	71
2.59.1	Function Documentation	71
2.59.1.1	free_map()	71
2.59.1.2	init_world()	71
2.60	server/init/server.c File Reference	72
2.60.1	Function Documentation	72
2.60.1.1	debug_no_reuse()	72
2.60.1.2	init_select()	72
2.60.1.3	init_server()	73
2.60.1.4	init_sock_address()	73
2.61	server/init/zappy.c File Reference	73
2.61.1	Function Documentation	74
2.61.1.1	free_zappy()	74
2.61.1.2	init_zappy()	74
2.62	server/main.c File Reference	74

2.62.1	Function Documentation	75
2.62.1.1	main()	75
2.62.1.2	zappy()	75
2.63	server/network/client/accept.c File Reference	75
2.63.1	Function Documentation	75
2.63.1.1	accept_client()	76
2.63.1.2	get_socket_client()	76
2.64	server/network/client/create.c File Reference	76
2.64.1	Function Documentation	76
2.64.1.1	create_client()	76
2.65	server/network/client/reset_client_timer.c File Reference	77
2.65.1	Function Documentation	77
2.65.1.1	reset_client_timer()	77
2.66	server/network/recv/recv_client_info.c File Reference	77
2.66.1	Function Documentation	77
2.66.1.1	recv_client_info()	78
2.67	server/network/rfds/give_player_to_client.c File Reference	78
2.67.1	Function Documentation	78
2.67.1.1	check_for_free_player()	78
2.67.1.2	give_player_to_client()	79
2.68	server/network/rfds/handle_client.c File Reference	79
2.68.1	Function Documentation	79
2.68.1.1	give_player_to_client_unknown()	79
2.68.1.2	handle_client()	80
2.68.1.3	handle_client_ai()	80
2.68.1.4	handle_client_graphic()	81
2.68.1.5	handle_client_unknown()	81
2.69	server/network/rfds/manage_rfds.c File Reference	81
2.69.1	Function Documentation	82
2.69.1.1	handle_new_connection()	82

2.69.1.2	manage_rfds()	82
2.70	server/network/rfds/parse_command.c File Reference	82
2.70.1	Function Documentation	82
2.70.1.1	add_str_to_command()	82
2.70.1.2	parse_command()	83
2.70.1.3	remove_endline_from_command()	83
2.71	server/network/select_server.c File Reference	83
2.71.1	Function Documentation	83
2.71.1.1	select_server()	83
2.72	server/network/send/add_to_init_graph_response.c File Reference	84
2.72.1	Function Documentation	84
2.72.1.1	add_to_init_graph_response()	84
2.73	server/network/send/respond_to_client.c File Reference	84
2.73.1	Function Documentation	85
2.73.1.1	respond_to_client()	85
2.74	server/network/send/send_info_to_graphical.c File Reference	85
2.74.1	Function Documentation	85
2.74.1.1	send_info_to_graphical()	85
2.75	server/network/send/send_init_info_to_graph.c File Reference	86
2.75.1	Function Documentation	86
2.75.1.1	send_init_info_to_graph()	86
2.76	server/network/wfds/manage_wfds.c File Reference	86
2.76.1	Function Documentation	87
2.76.1.1	handle_client_wfds()	87
2.76.1.2	manage_wfds()	87
2.77	server/network/wfds/manage_wfds_ai.c File Reference	87
2.77.1	Function Documentation	88
2.77.1.1	handle_client_ai_wfds()	88
2.77.1.2	init_serv_info_buffer()	88
2.78	server/network/wfds/manage_wfds_graph.c File Reference	89

2.78.1	Function Documentation	89
2.78.1.1	handle_client_graph_wfds()	89
2.79	server/network/wfds/set_wfds.c File Reference	89
2.79.1	Function Documentation	90
2.79.1.1	if_not_set_wfds()	90
2.79.1.2	if_set_wfds()	90
2.79.1.3	set_wfds()	90
2.80	server/run_zappy.c File Reference	90
2.80.1	Function Documentation	91
2.80.1.1	run_zappy()	91
2.81	server/tools/array_tools.c File Reference	91
2.81.1	Function Documentation	91
2.81.1.1	get_array_size()	91
2.82	server/tools/buffer.c File Reference	92
2.82.1	Function Documentation	92
2.82.1.1	add_str_to_buffer()	92
2.82.1.2	reset_buffer()	92
2.83	server/tools/error/int.c File Reference	93
2.83.1	Function Documentation	93
2.83.1.1	error()	93
2.83.1.2	error_close()	93
2.84	server/tools/error/print.c File Reference	93
2.84.1	Function Documentation	94
2.84.1.1	error_print()	94
2.84.1.2	error_print_ptr()	94
2.85	server/tools/error/ptr.c File Reference	94
2.85.1	Function Documentation	94
2.85.1.1	error_close_ptr()	94
2.85.1.2	error_ptr()	95
2.86	server/tools/usage.c File Reference	95
2.86.1	Function Documentation	95
2.86.1.1	usage()	95

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

server/main.c	74
server/run_zappy.c	90
server/circular_buffer/add_to_buffer.c	3
server/circular_buffer/has_command.c	4
server/circular_buffer/init_buffer.c	5
server/circular_buffer/read_from_buffer.c	6
server/clock/check_for_timeouts.c	7
server/clock/clock_tick.c	7
server/game/commands/broadcast.c	9
server/game/commands/command_list.c	12
server/game/commands/connect_nbr.c	12
server/game/commands/eject.c	13
server/game/commands/fork.c	14
server/game/commands/incantation.c	15
server/game/commands/inventory.c	18
server/game/commands/look.c	18
server/game/commands/look_tile.c	20
server/game/commands/move.c	21
server/game/commands/take_set.c	23
server/game/inventory/check_ritual_resources.c	24
server/game/inventory/get_resource.c	25
server/game/inventory/transfer_item.c	26
server/game/log/hatch_log.c	28
server/game/log/incantation_log.c	28
server/game/log/kill_log.c	29
server/game/log/move_log.c	30
server/game/log/spawn_log.c	31
server/game/log/take_log.c	32
server/game/map/add_player_to_tile.c	33
server/game/map/get_dir.c	34
server/game/map/get_nbr_players.c	35
server/game/map/get_player.c	38
server/game/map/get_tile.c	39
server/game/map/move_player_to_tile.c	41
server/game/map/remove_player_from_tile.c	43

server/game/map/remove_resources_from_tile.c	44
server/game/map/spawn_food.c	44
server/game/player/add.c	46
server/game/player/append.c	48
server/game/player/check.c	49
server/game/player/delete.c	50
server/game/player/get.c	54
server/game/team/add_player.c	56
server/game/team/get_nbr_players.c	36
server/game/team/get_player.c	38
server/game/team/get_team.c	57
server/game/team/remove_player.c	57
server/game/update/check_if_game_won.c	58
server/game/update/execute_player_commands.c	59
server/game/update/set_player_command.c	60
server/game/update/update_game.c	62
server/get_args/get_args.c	63
server/get_args/options_func.c	65
server/get_args/options_names_func.c	66
server/init/server.c	72
server/init/zappy.c	73
server/init/game/game.c	67
server/init/game/link_tiles.c	68
server/init/game/teams.c	69
server/init/game/world.c	71
server/network/select_server.c	83
server/network/client/accept.c	75
server/network/client/add.c	48
server/network/client/create.c	76
server/network/client/delete.c	52
server/network/client/get.c	55
server/network/client/reset_client_timer.c	77
server/network/recv/recv_client_info.c	77
server/network/rfds/give_player_to_client.c	78
server/network/rfds/handle_client.c	79
server/network/rfds/manage_rfds.c	81
server/network/rfds/parse_command.c	82
server/network/send/add_to_init_graph_response.c	84
server/network/send/respond_to_client.c	84
server/network/send/send_info_to_graphical.c	85
server/network/send/send_init_info_to_graph.c	86
server/network/wfds/manage_wfds.c	86
server/network/wfds/manage_wfds_ai.c	87
server/network/wfds/manage_wfds_graph.c	89
server/network/wfds/set_wfds.c	89
server/tools/array_tools.c	91
server/tools/buffer.c	92
server/tools/usage.c	95
server/tools/error/int.c	93
server/tools/error/print.c	93
server/tools/error/ptr.c	94

Chapter 2

File Documentation

2.1 server/circular_buffer/add_to_buffer.c File Reference

```
#include "circular_buffer.h"
```

Functions

- int [get_len_left](#) (c_buffer_t *buffer)
Get free space in buffer.
- bool [can_fit_in_buffer](#) (int len, c_buffer_t *buffer)
Check if given data will fit in the buffer.
- int [add_to_buffer](#) (char *str, c_buffer_t *buffer)
Add data to the buffer.

2.1.1 Function Documentation

2.1.1.1 add_to_buffer()

```
int add_to_buffer (  
    char * str,  
    c_buffer_t * buffer )
```

Add data to the buffer.

Parameters

<i>str</i>	
<i>buffer</i>	

Returns

length of the added content, ADD_FAIL on error

Definition at line 31 of file add_to_buffer.c.

2.1.1.2 can_fit_in_buffer()

```
bool can_fit_in_buffer (
    int len,
    c_buffer_t * buffer )
```

Check if given data will fit in the buffer.

Parameters

<i>len</i>	
<i>buffer</i>	

Returns

true if small enough

Definition at line 22 of file add_to_buffer.c.

2.1.1.3 get_len_left()

```
int get_len_left (
    c_buffer_t * buffer )
```

Get free space in buffer.

Parameters

<i>buffer</i>	
---------------	--

Returns

free space left

Definition at line 13 of file add_to_buffer.c.

2.2 server/circular_buffer/has_command.c File Reference

```
#include "circular_buffer.h"
```

Functions

- bool [has_command](#) (c_buffer_t *buffer)
Check if the buffer contain a valid command.

2.2.1 Function Documentation

2.2.1.1 has_command()

```
bool has_command (
    c_buffer_t * buffer )
```

Check if the buffer contain a valid command.

Parameters

<i>buffer</i>	
---------------	--

Returns

Definition at line 13 of file has_command.c.

2.3 server/circular_buffer/init_buffer.c File Reference

```
#include "circular_buffer.h"
```

Functions

- void [init_c_buffer](#) (c_buffer_t *buffer)
Setup default buffer parameters.

2.3.1 Function Documentation

2.3.1.1 init_c_buffer()

```
void init_c_buffer (
    c_buffer_t * buffer )
```

Setup default buffer parameters.

Parameters

<i>buffer</i>	
---------------	--

Definition at line 12 of file init_buffer.c.

2.4 server/circular_buffer/read_from_buffer.c File Reference

```
#include "circular_buffer.h"
```

Functions

- bool [is_buf_readable](#) (c_buffer_t *buffer)
Check if the buffer contain something.
- char * [read_from_buffer](#) (c_buffer_t *buffer)
Check if not empty and get a valid command from it.

2.4.1 Function Documentation

2.4.1.1 is_buf_readable()

```
bool is_buf_readable (  
    c_buffer_t * buffer )
```

Check if the buffer contain something.

Parameters

<i>buffer</i>	
---------------	--

Returns

true if not empty

Definition at line 13 of file read_from_buffer.c.

2.4.1.2 read_from_buffer()

```
char* read_from_buffer (  
    c_buffer_t * buffer )
```

Check if not empty and get a valid command from it.

Parameters

<i>buffer</i>	
---------------	--

Returns

a valid command

Definition at line 23 of file read_from_buffer.c.

2.5 server/clock/check_for_timeouts.c File Reference

```
#include "zappy.h"
```

Functions

- void [check_for_timeouts](#) (zappy_t *[zappy](#))
Disconnect a client if it doesn't respond early.

2.5.1 Function Documentation

2.5.1.1 check_for_timeouts()

```
void check_for_timeouts (  
    zappy_t * zappy )
```

Disconnect a client if it doesn't respond early.

Parameters

<i>zappy</i>	
--------------	--

Definition at line 12 of file check_for_timeouts.c.

2.6 server/clock/clock_tick.c File Reference

```
#include "game/log.h"  
#include "zappy.h"
```

Functions

- void [clock_tick_clients](#) (client_t *client, clock_t time_diff)
Update client timeout.
- int [clock_tick_player_ttl](#) (player_t **player, game_t *game, int ttf)
Check if player have enough and otherwise kill it.
- void [clock_tick_players](#) (game_t *game, clock_t time_diff, double f)
Loop over all players and update their food.
- void [clock_tick](#) (zappy_t *zappy, clock_t time_start)
Update game time.

2.6.1 Function Documentation

2.6.1.1 clock_tick()

```
void clock_tick (
    zappy_t * zappy,
    clock_t time_start )
```

Update game time.

Parameters

<i>zappy</i>	
<i>time_start</i>	

Definition at line 80 of file clock_tick.c.

2.6.1.2 clock_tick_clients()

```
void clock_tick_clients (
    client_t * client,
    clock_t time_diff )
```

Update client timeout.

Parameters

<i>client</i>	
<i>time_diff</i>	

Definition at line 14 of file clock_tick.c.

2.6.1.3 clock_tick_player_ttl()

```
int clock_tick_player_ttl (
    player_t ** player,
    game_t * game,
    int ttf )
```

Check if player have enough and otherwise kill it.

Parameters

<i>player</i>	
<i>game</i>	
<i>ttf</i>	

Returns

1 if the player died, 0 otherwise

Definition at line 29 of file clock_tick.c.

2.6.1.4 clock_tick_players()

```
void clock_tick_players (
    game_t * game,
    clock_t time_diff,
    double f )
```

Loop over all players and update their food.

Parameters

<i>game</i>	
<i>time_diff</i>	
<i>f</i>	

Definition at line 56 of file clock_tick.c.

2.7 server/game/commands/broadcast.c File Reference

```
#include "game/commands.h"
#include "macros.h"
#include <math.h>
```

Functions

- int `convert_angle_to_num` (double angle)
Convert an angle in degree to the possible tile output.
- double `get_angle` (vec2i_t target, vec2i_t origin)
Get an angle from two vectors.
- vec2i_t `get_origin` (dir_t dir, vec2i_t *src)
Create a vector from a position and a direction.
- char `get_traj` (player_t *source_player, player_t *dest_player, world_t *world)
- int `cmd_broadcast` (char *cmd, player_t *player, game_t *game)
Send a message to all player, with the direction the message is coming from.

2.7.1 Function Documentation

2.7.1.1 cmd_broadcast()

```
int cmd_broadcast (
    char * cmd,
    player_t * player,
    game_t * game )
```

Send a message to all player, with the direction the message is coming from.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

status of execution

Definition at line 92 of file broadcast.c.

2.7.1.2 convert_angle_to_num()

```
int convert_angle_to_num (
    double angle )
```

Convert an angle in degree to the possible tile output.

Parameters

<i>angle</i>	
--------------	--

Returns

0 to 8

Definition at line 16 of file broadcast.c.

2.7.1.3 get_angle()

```
double get_angle (
    vec2i_t target,
    vec2i_t origin )
```

Get an angle from two vectors.

Parameters

<i>target</i>	
<i>origin</i>	

Returns

actual angle in degree

Definition at line 41 of file broadcast.c.

2.7.1.4 get_origin()

```
vec2i_t get_origin (
    dir_t dir,
    vec2i_t * src )
```

Create a vector from a position and a direction.

Parameters

<i>dir</i>	
<i>src</i>	

Returns

the vector

Definition at line 56 of file broadcast.c.

2.7.1.5 get_traj()

```
char get_traj (
    player_t * source_player,
    player_t * dest_player,
    world_t * world )
```

Definition at line 72 of file broadcast.c.

2.8 server/game/commands/command_list.c File Reference

```
#include "game/commands.h"
```

Variables

- const struct s_command [command_list](#) []

2.8.1 Variable Documentation

2.8.1.1 command_list

```
const struct s_command command_list[]
```

Initial value:

```
= {
    {.name = "Forward", .cmd_ptr = &cmd_forward, 7},
    {.name = "Right", .cmd_ptr = &cmd_right, 7},
    {.name = "Left", .cmd_ptr = &cmd_left, 7},
    {.name = "Look", .cmd_ptr = &cmd_look, 7},
    {.name = "Inventory", .cmd_ptr = &cmd_inventory, 1},
    {.name = "Broadcast", .cmd_ptr = &cmd_broadcast, 7},
    {.name = "Connect_nbr", .cmd_ptr = &cmd_connect_nbr, 0},
    {.name = "Fork", .cmd_ptr = &cmd_fork, 42},
    {.name = "Eject", .cmd_ptr = &cmd_eject, 7},
    {.name = "Take", .cmd_ptr = &cmd_take, 7},
    {.name = "Set", .cmd_ptr = &cmd_set, 7},
    {.name = "Incantation", .cmd_ptr = &cmd_incantation, 300},
    {.name = NULL}
}
```

Definition at line 10 of file command_list.c.

2.9 server/game/commands/connect_nbr.c File Reference

```
#include "game/commands.h"
#include "macros.h"
```

Functions

- int [cmd_connect_nbr](#) (char *cmd, player_t *player, game_t *game)
Return the number of unused players in the game.

2.9.1 Function Documentation

2.9.1.1 cmd_connect_nbr()

```
int cmd_connect_nbr (
    char * cmd,
    player_t * player,
    game_t * game )
```

Return the number of unused players in the game.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 17 of file connect_nbr.c.

2.10 server/game/commands/eject.c File Reference

```
#include "game/commands.h"
#include "macros.h"
```

Functions

- char * [dir_to_str](#) (dir_t dir)
- bool [init_eject_msg](#) (player_t *player, dir_t dir)
- int [cmd_eject](#) (char *cmd, player_t *player, game_t *game)
Eject a player from his tile.

2.10.1 Function Documentation

2.10.1.1 cmd_eject()

```
int cmd_eject (
    char * cmd,
    player_t * player,
    game_t * game )
```

Eject a player from his tile.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 38 of file eject.c.

2.10.1.2 dir_to_str()

```
char* dir_to_str (
    dir_t dir )
```

Definition at line 12 of file eject.c.

2.10.1.3 init_eject_msg()

```
bool init_eject_msg (
    player_t * player,
    dir_t dir )
```

Definition at line 21 of file eject.c.

2.11 server/game/commands/fork.c File Reference

```
#include "game/commands.h"
#include "macros.h"
```

Functions

- int [cmd_fork](#) (char *cmd, player_t *player, game_t *game)
Fork a player to a egg.

2.11.1 Function Documentation

2.11.1.1 cmd_fork()

```
int cmd_fork (
    char * cmd,
    player_t * player,
    game_t * game )
```

Fork a player to a egg.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 17 of file fork.c.

2.12 server/game/commands/incantation.c File Reference

```
#include "game/log.h"
#include "game/commands.h"
#include "game/map.h"
#include "macros.h"
```

Functions

- bool [check_if_ritual_req_met](#) (const ritual_data_t *req, player_t *player)
Check if the given player meet the given requirements.
- void [upgrade_ritual_players](#) (player_t *tile_players, int last_lvl)
Run over all players to increase their level after a successful incantation.
- void [send_ko_to_ritual_players](#) (player_t *tile_players, int cur_level)
Send ko message to players who failed an incantation.
- int [cmd_incantation](#) (char *cmd, player_t *player, game_t *game)
Allow player to incant.

Variables

- const ritual_data_t [ritual_data](#) []

2.12.1 Function Documentation

2.12.1.1 `check_if_ritual_req_met()`

```
bool check_if_ritual_req_met (
    const ritual_data_t * req,
    player_t * player )
```

Check if the given player meet the given requirements.

Parameters

<i>req</i>	
<i>player</i>	

Returns

true or false

Definition at line 33 of file incantation.c.

2.12.1.2 `cmd_incantation()`

```
int cmd_incantation (
    char * cmd,
    player_t * player,
    game_t * game )
```

Allow player to incant.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

status

Definition at line 85 of file incantation.c.

2.12.1.3 send_ko_to_ritual_players()

```
void send_ko_to_ritual_players (
    player_t * tile_players,
    int cur_level )
```

Send ko message to players who failed an incantation.

Parameters

<i>tile_players</i>	
<i>cur_level</i>	

Definition at line 71 of file incantation.c.

2.12.1.4 upgrade_ritual_players()

```
void upgrade_ritual_players (
    player_t * tile_players,
    int last_lvl )
```

Run over all players to increase their level after a successful incantation.

Parameters

<i>tile_players</i>	
<i>last_lvl</i>	

Definition at line 52 of file incantation.c.

2.12.2 Variable Documentation

2.12.2.1 ritual_data

```
const ritual_data_t ritual_data[]
```

Initial value:

```
= {
    {.level = 1, .nb_players = 1, .inv = (inventory_t) {0, 1, 0, 0, 0, 0,
    0}},
    {.level = 2, .nb_players = 2, .inv = (inventory_t) {0, 1, 1, 1, 0, 0,
    0}},
    {.level = 3, .nb_players = 2, .inv = (inventory_t) {0, 2, 0, 1, 0, 2,
    0}},
    {.level = 4, .nb_players = 4, .inv = (inventory_t) {0, 1, 1, 2, 0, 1,
    0}},
    {.level = 5, .nb_players = 4, .inv = (inventory_t) {0, 1, 2, 1, 3, 0,
    0}},
    {.level = 6, .nb_players = 6, .inv = (inventory_t) {0, 1, 2, 3, 0, 1,
    0}},
    {.level = 7, .nb_players = 6, .inv = (inventory_t) {0, 2, 2, 2, 2, 2,
    1}}}
```

Definition at line 13 of file incantation.c.

2.13 server/game/commands/inventory.c File Reference

```
#include "macros.h"
#include "game/commands.h"
```

Functions

- int [cmd_inventory](#) (char *cmd, player_t *player, game_t *game)
send the player's inventory

2.13.1 Function Documentation

2.13.1.1 cmd_inventory()

```
int cmd_inventory (
    char * cmd,
    player_t * player,
    game_t * game )
```

send the player's inventory

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

status

Definition at line 16 of file inventory.c.

2.14 server/game/commands/look.c File Reference

```
#include "macros.h"
#include "errors.h"
#include "game/commands.h"
```


Functions

- int [fill_resp_look_row](#) (buffer_t *buf, tile_t *first, dir_t dir_look, int i)
- int [fill_resp_look](#) (buffer_t *buf, player_t *player)
Format output in a readable look server response.
- int [cmd_look](#) (char *cmd, player_t *player, game_t *game)
Get what is in the field of view of the player.

2.14.1 Function Documentation

2.14.1.1 cmd_look()

```
int cmd_look (  
    char * cmd,  
    player_t * player,  
    game_t * game )
```

Get what is in the field of view of the player.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

status

Definition at line 67 of file look.c.

2.14.1.2 fill_resp_look()

```
int fill_resp_look (  
    buffer_t * buf,  
    player_t * player )
```

Format output in a readable look server response.

Parameters

<i>buf</i>	
<i>player</i>	

Returns

status

Definition at line 42 of file look.c.

2.14.1.3 fill_resp_look_row()

```
int fill_resp_look_row (
    buffer_t * buf,
    tile_t * first,
    dir_t dir_look,
    int i )
```

Definition at line 24 of file look.c.

2.15 server/game/commands/look_tile.c File Reference

```
#include "macros.h"
#include "game/commands.h"
#include "errors.h"
```

Functions

- int [look_print](#) (buffer_t *buffer, uint16_t content, char *name)
- int [look_print_player](#) (buffer_t *buffer, tile_t *tile)

Fill the buffer with the corresponding number of "players" string.

- int [look_tile](#) (buffer_t *buffer, tile_t *tile)

Fill the buffer with the content of the given tile.

2.15.1 Function Documentation

2.15.1.1 look_print()

```
int look_print (
    buffer_t * buffer,
    uint16_t content,
    char * name )
```

Definition at line 12 of file look_tile.c.

2.15.1.2 look_print_player()

```
int look_print_player (
    buffer_t * buffer,
    tile_t * tile )
```

Fill the buffer with the corresponding number of "players" string.

Parameters

<i>buffer</i>	
<i>tile</i>	

Returns

Definition at line 30 of file look_tile.c.

2.15.1.3 look_tile()

```
int look_tile (
    buffer_t * buffer,
    tile_t * tile )
```

Fill the buffer with the content of the given tile.

Parameters

<i>buffer</i>	
<i>tile</i>	

Returns

Definition at line 50 of file look_tile.c.

2.16 server/game/commands/move.c File Reference

```
#include "game/game.h"
#include "game/log.h"
#include "game/commands.h"
#include "macros.h"
```

Functions

- int [cmd_forward](#) (char *cmd, player_t *player, game_t *game)
Allow the player to move forward.
- int [cmd_right](#) (char *cmd, player_t *player, game_t *game)
Allow the player to move right.
- int [cmd_left](#) (char *cmd, player_t *player, game_t *game)
Allow the player to move left.

2.16.1 Function Documentation

2.16.1.1 cmd_forward()

```
int cmd_forward (
    char * cmd,
    player_t * player,
    game_t * game )
```

Allow the player to move forward.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 18 of file move.c.

2.16.1.2 cmd_left()

```
int cmd_left (
    char * cmd,
    player_t * player,
    game_t * game )
```

Allow the player to move left.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 63 of file move.c.

2.16.1.3 cmd_right()

```
int cmd_right (
    char * cmd,
    player_t * player,
    game_t * game )
```

Allow the player to move right.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 34 of file move.c.

2.17 server/game/commands/take_set.c File Reference

```
#include "macros.h"
#include "game/log.h"
#include "game/commands.h"
```

Functions

- int [cmd_take](#) (char *cmd, player_t *player, game_t *game)
Remove an item from the ground and add it to the player inventory.
- int [cmd_set](#) (char *cmd, player_t *player, game_t *game)
Remove an item from the player inventory and add it to the ground.

2.17.1 Function Documentation

2.17.1.1 cmd_set()

```
int cmd_set (
    char * cmd,
    player_t * player,
    game_t * game )
```

Remove an item from the player inventory and add it to the ground.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 45 of file take_set.c.

2.17.1.2 cmd_take()

```
int cmd_take (
    char * cmd,
    player_t * player,
    game_t * game )
```

Remove an item from the ground and add it to the player inventory.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 17 of file take_set.c.

2.18 server/game/inventory/check_ritual_resources.c File Reference

```
#include "game/inventory.h"
```

Functions

- bool [check_ritual_resources](#) (const inventory_t *tile_inv, const inventory_t *ritual_req)
Check if the player have the necessary items for a ritual.

2.18.1 Function Documentation

2.18.1.1 check_ritual_resources()

```
bool check_ritual_resources (
    const inventory_t * tile_inv,
    const inventory_t * ritual_req )
```

Check if the player have the necessary items for a ritual.

Parameters

<i>tile_inv</i>	
<i>ritual_req</i>	

Returns

Definition at line 14 of file check_ritual_resources.c.

2.19 server/game/inventory/get_resource.c File Reference

```
#include "game/game.h"
#include "game/inventory.h"
```

Functions

- resource_t [get_random_resource](#) (void)
Get a random resource between the all available ones.
- resource_t [get_resource_from_str](#) (char *str)
Convert a string to his resource_t equivalent.

2.19.1 Function Documentation

2.19.1.1 get_random_resource()

```
resource_t get_random_resource (
    void )
```

Get a random resource between the all available ones.

Returns

a random food

Definition at line 13 of file get_resource.c.

2.19.1.2 `get_resource_from_str()`

```
resource_t get_resource_from_str (
    char * str )
```

Convert a string to his resource_t equivalent.

Parameters

<i>str</i>	
------------	--

Returns

the resource

Definition at line 27 of file `get_resource.c`.

2.20 `server/game/inventory/transfer_item.c` File Reference

```
#include "game/inventory.h"
```

Functions

- bool [add_resource_to_inventory](#) (inventory_t *inventory, resource_t resource)
Add the resource in the given inventory.
- bool [del_resource_from_inventory](#) (inventory_t *inventory, resource_t resource)
Relive the resource from the given inventory.
- bool [transfer_item](#) (inventory_t *inv_from, inventory_t *inv_to, resource_t res)
Move a resource between to inventories.

2.20.1 Function Documentation

2.20.1.1 `add_resource_to_inventory()`

```
bool add_resource_to_inventory (
    inventory_t * inventory,
    resource_t resource )
```

Add the resource in the given inventory.

Parameters

<i>inventory</i>	
<i>resource</i>	

Returns

Definition at line 14 of file transfer_item.c.

2.20.1.2 del_resource_from_inventory()

```
bool del_resource_from_inventory (
    inventory_t * inventory,
    resource_t resource )
```

Relive the resource from the given inventory.

Parameters

<i>inventory</i>	
<i>resource</i>	

Returns

true if success, false otherwise

Definition at line 31 of file transfer_item.c.

2.20.1.3 transfer_item()

```
bool transfer_item (
    inventory_t * inv_from,
    inventory_t * inv_to,
    resource_t res )
```

Move a resource between to inventories.

Parameters

<i>inv_from</i>	
<i>inv_to</i>	
<i>res</i>	

Returns

true if success, false otherwise

Definition at line 49 of file transfer_item.c.

2.21 server/game/log/hatch_log.c File Reference

```
#include "game/log.h"
```

Functions

- int [hatch_log](#) (player_t *player, buffer_t *game_log)
Send hatch to the graphic client.

2.21.1 Function Documentation

2.21.1.1 hatch_log()

```
int hatch_log (  
    player_t * player,  
    buffer_t * game_log )
```

Send hatch to the graphic client.

Parameters

<i>player</i>	
<i>game_log</i>	

Returns

Definition at line 14 of file hatch_log.c.

2.22 server/game/log/incantation_log.c File Reference

```
#include "game/log.h"
```

Functions

- int [incantation_start_log](#) (vec2i_t *pos, buffer_t *game_log)
Send a notification that an incantation has started to the graphic.
- int [incantation_end_log](#) (vec2i_t *pos, bool res, buffer_t *game_log)
Send a notification that an incantation ended to the graphic.

2.22.1 Function Documentation

2.22.1.1 incantation_end_log()

```
int incantation_end_log (
    vec2i_t * pos,
    bool res,
    buffer_t * game_log )
```

Send a notification that an incantation ended to the graphic.

Parameters

<i>pos</i>	
<i>res</i>	
<i>game_log</i>	

Returns

Definition at line 27 of file incantation_log.c.

2.22.1.2 incantation_start_log()

```
int incantation_start_log (
    vec2i_t * pos,
    buffer_t * game_log )
```

Send a notification that an incantation has started to the graphic.

Parameters

<i>pos</i>	
<i>game_log</i>	

Returns

Definition at line 14 of file incantation_log.c.

2.23 server/game/log/kill_log.c File Reference

```
#include "game/log.h"
```

Functions

- int [kill_log](#) (player_t *player, buffer_t *game_log)
Send a notification to the graphic that a player died.

2.23.1 Function Documentation

2.23.1.1 kill_log()

```
int kill_log (  
    player_t * player,  
    buffer_t * game_log )
```

Send a notification to the graphic that a player died.

Parameters

<i>player</i>	
<i>game_log</i>	

Returns

Definition at line 14 of file kill_log.c.

2.24 server/game/log/move_log.c File Reference

```
#include "game/log.h"
```

Functions

- int [forward_log](#) (player_t *player, buffer_t *game_log)
Send a notification to the graphic that a player moved forward.
- int [dir_log](#) (player_t *player, buffer_t *game_log)

2.24.1 Function Documentation

2.24.1.1 `dir_log()`

```
int dir_log (
    player_t * player,
    buffer_t * game_log )
```

Definition at line 22 of file `move_log.c`.

2.24.1.2 `forward_log()`

```
int forward_log (
    player_t * player,
    buffer_t * game_log )
```

Send a notification to the graphic that a player moved forward.

Parameters

<i>player</i>	
<i>game_log</i>	

Returns

Definition at line 14 of file `move_log.c`.

2.25 server/game/log/spawn_log.c File Reference

```
#include "game/log.h"
```

Functions

- bool [spawn_player_log](#) (player_t *player, buffer_t *game_log)
Tell the graphic a new player arrived.
- bool [spawn_resource_log](#) (vec2i_t *pos, resource_t resource, buffer_t *game_log)
Tell the graphic a new resource spawned.

2.25.1 Function Documentation

2.25.1.1 `spawn_player_log()`

```
bool spawn_player_log (
    player_t * player,
    buffer_t * game_log )
```

Tell the graphic a new player arrived.

Parameters

<i>player</i>	
<i>game_log</i>	

Returns

Definition at line 14 of file spawn_log.c.

2.25.1.2 spawn_resource_log()

```
bool spawn_resource_log (
    vec2i_t * pos,
    resource_t resource,
    buffer_t * game_log )
```

Tell the graphic a new resource spawned.

Parameters

<i>pos</i>	
<i>resource</i>	
<i>game_log</i>	

Returns

Definition at line 39 of file spawn_log.c.

2.26 server/game/log/take_log.c File Reference

```
#include "game/log.h"
```

Functions

- int [take_log](#) (int player_id, resource_t res, buffer_t *game_log)
Tell the graphic to take a resource.
- int [take_log_inventory](#) (const inventory_t *inv, int player_id, buffer_t *game_log)

2.26.1 Function Documentation

2.26.1.1 take_log()

```
int take_log (
    int player_id,
    resource_t res,
    buffer_t * game_log )
```

Tell the graphic to take a resource.

Parameters

<i>player_id</i>	
<i>res</i>	
<i>game_log</i>	

Returns

Definition at line 15 of file take_log.c.

2.26.1.2 take_log_inventory()

```
int take_log_inventory (
    const inventory_t * inv,
    int player_id,
    buffer_t * game_log )
```

Definition at line 23 of file take_log.c.

2.27 server/game/map/add_player_to_tile.c File Reference

```
#include "game/player.h"
#include "game/map.h"
```

Functions

- bool [append_to_tile_player_list](#) (tile_t *tile, player_t *player)
Push a player into the tile's players linked list.

2.27.1 Function Documentation

2.27.1.1 `append_to_tile_player_list()`

```
bool append_to_tile_player_list (
    tile_t * tile,
    player_t * player )
```

Push a player into the tile's players linked list.

Parameters

<i>tile</i>	
<i>player</i>	

Returns

true or false, depending on success

Definition at line 15 of file `add_player_to_tile.c`.

2.28 `server/game/map/get_dir.c` File Reference

```
#include "game/map.h"
```

Functions

- `dir_t get_dir_to_left` (`dir_t dir`)
Get the relative left direction from a given one.
- `dir_t get_dir_to_right` (`dir_t dir`)
Get the relative right direction from a given.

2.28.1 Function Documentation

2.28.1.1 `get_dir_to_left()`

```
dir_t get_dir_to_left (
    dir_t dir )
```

Get the relative left direction from a given one.

Parameters

<i>dir</i>	
------------	--

Returns

a direction

Definition at line 13 of file get_dir.c.

2.28.1.2 get_dir_to_right()

```
dir_t get_dir_to_right (
    dir_t dir )
```

Get the relative right direction from a given.

Parameters

<i>dir</i>	
------------	--

Returns

a direction

Definition at line 28 of file get_dir.c.

2.29 server/game/map/get_nbr_players.c File Reference

```
#include "game/player.h"
#include "game/map.h"
```

Functions

- int [get_nbr_players_of_lvl_on_tile](#) (tile_t *tile, int lvl)
Count players with a given level.
- int [get_nbr_players_on_tile](#) (tile_t *tile)
Count players.

2.29.1 Function Documentation

2.29.1.1 `get_nbr_players_of_lvl_on_tile()`

```
int get_nbr_players_of_lvl_on_tile (
    tile_t * tile,
    int lvl )
```

Count players with a given level.

Parameters

<i>tile</i>	
<i>lvl</i>	

Returns

count

Definition at line 15 of file `get_nbr_players.c`.

2.29.1.2 `get_nbr_players_on_tile()`

```
int get_nbr_players_on_tile (
    tile_t * tile )
```

Count players.

Parameters

<i>tile</i>	
-------------	--

Returns

count

Definition at line 28 of file `get_nbr_players.c`.

2.30 `server/game/team/get_nbr_players.c` File Reference

```
#include "game/team.h"
```

Functions

- int [get_team_nbr_players](#) (team_t *team)
Count the number of players in a given team.
- int [get_team_nbr_players_free](#) (team_t *team)
Count how much players free from a client in a given team.
- int [get_team_nbr_players_not_free](#) (team_t *team)
Count how much players with client in a given team.

2.30.1 Function Documentation

2.30.1.1 get_team_nbr_players()

```
int get_team_nbr_players (
    team_t * team )
```

Count the number of players in a given team.

Parameters

<i>team</i>	
-------------	--

Returns

count

Definition at line 13 of file get_nbr_players.c.

2.30.1.2 get_team_nbr_players_free()

```
int get_team_nbr_players_free (
    team_t * team )
```

Count how much players free from a client in a given team.

Parameters

<i>team</i>	
-------------	--

Returns

count

Definition at line 27 of file get_nbr_players.c.

2.30.1.3 get_team_nbr_players_not_free()

```
int get_team_nbr_players_not_free (
    team_t * team )
```

Count how much players with client in a given team.

Parameters

<i>team</i>	
-------------	--

Returns

count

Definition at line 43 of file `get_nbr_players.c`.

2.31 `server/game/map/get_player.c` File Reference

```
#include "game/player.h"
#include "game/map.h"
```

Functions

- `player_t * get_last_tile_player (tile_t *tile)`
Get the last player in the linked list of players of a tile.

2.31.1 Function Documentation

2.31.1.1 `get_last_tile_player()`

```
player_t* get_last_tile_player (
    tile_t * tile )
```

Get the last player in the linked list of players of a tile.

Parameters

<i>tile</i>	
-------------	--

Returns

The player, or null if tile is empty

Definition at line 14 of file `get_player.c`.

2.32 `server/game/team/get_player.c` File Reference

```
#include "game/team.h"
```

Functions

- `player_t * get_last_team_player (team_t *team)`
Get the last player in a given team.

2.32.1 Function Documentation

2.32.1.1 get_last_team_player()

```
player_t* get_last_team_player (
    team_t * team )
```

Get the last player in a given team.

Parameters

<i>team</i>	
-------------	--

Returns

a pointer to the player

Definition at line 13 of file `get_player.c`.

2.33 server/game/map/get_tile.c File Reference

```
#include "game/map.h"
```

Functions

- `tile_t * get_tile (world_t *world, vec2i_t *pos)`
Get the tile at the given position in the world.
- `tile_t * get_tile_in_dir (tile_t *tile, dir_t dir)`
Get the next tile in the given direction.
- `tile_t * get_random_tile (world_t *world)`
Pick a random tile in the world.

2.33.1 Function Documentation

2.33.1.1 get_random_tile()

```
tile_t* get_random_tile (
    world_t * world )
```

Pick a random tile in the world.

Parameters

<i>world</i>	
--------------	--

Returns

a pointer to the tile

Definition at line 47 of file `get_tile.c`.

2.33.1.2 `get_tile()`

```
tile_t* get_tile (
    world_t * world,
    vec2i_t * pos )
```

Get the tile at the given position in the world.

Parameters

<i>world</i>	
<i>pos</i>	

Returns

a pointer to the tile or null if not found

Definition at line 14 of file `get_tile.c`.

2.33.1.3 `get_tile_in_dir()`

```
tile_t* get_tile_in_dir (
    tile_t * tile,
    dir_t dir )
```

Get the next tile in the given direction.

Parameters

<i>tile</i>	
<i>dir</i>	

Returns

the found tile, or null if invalid direction

Definition at line 28 of file get_tile.c.

2.34 server/game/map/move_player_to_tile.c File Reference

```
#include "game/player.h"
```

Functions

- bool [move_player_to_tile](#) (player_t *player, tile_t *tile)
Move the given player to the new tile.
- bool [move_player_in_dir](#) (player_t *player, dir_t dir)
Move the given player to the next tile in the given direction.

2.34.1 Function Documentation**2.34.1.1 move_player_in_dir()**

```
bool move_player_in_dir (
    player_t * player,
    dir_t dir )
```

Move the given player to the next tile in the given direction.

Parameters

<i>player</i>	
<i>dir</i>	

Returns

true or false, depending on success

Definition at line 32 of file move_player_to_tile.c.

2.34.1.2 move_player_to_tile()

```
bool move_player_to_tile (
    player_t * player,
    tile_t * tile )
```

Move the given player to the new tile.

Parameters

<i>player</i>	
<i>tile</i>	

Returns

true or false, depending on success

Definition at line 14 of file move_player_to_tile.c.

2.35 server/game/map/remove_player_from_tile.c File Reference

```
#include "game/player.h"  
#include "game/map.h"
```

Functions

- bool [remove_player_from_tile](#) (player_t *player, tile_t *tile)
Remove a player from a tile.

2.35.1 Function Documentation

2.35.1.1 remove_player_from_tile()

```
bool remove_player_from_tile (  
    player_t * player,  
    tile_t * tile )
```

Remove a player from a tile.

Parameters

<i>player</i>	
<i>tile</i>	

Returns

true or false, depending on success

Definition at line 15 of file remove_player_from_tile.c.

2.36 server/game/map/remove_resources_from_tile.c File Reference

```
#include "game/map.h"
```

Functions

- bool [remove_resources_from_tile](#) (const inventory_t *inv, tile_t *tile)
Remove resources from the given tile.

2.36.1 Function Documentation

2.36.1.1 remove_resources_from_tile()

```
bool remove_resources_from_tile (
    const inventory_t * inv,
    tile_t * tile )
```

Remove resources from the given tile.

Parameters

<i>inv</i>	
<i>tile</i>	

Returns

Definition at line 14 of file remove_resources_from_tile.c.

2.37 server/game/map/spawn_food.c File Reference

```
#include <stdio.h>
#include "game/log.h"
#include "game/game.h"
```

Functions

- bool [spawn_resource_at_pos](#) (world_t *map, vec2i_t *pos, resource_t resource)
Spawn a resource at the given pos in the world.
- void [init_random_pos](#) (vec2i_t *pos, int width, int height)
Fill a vector with random value, boundaries are width and height.
- void [spawn_resource](#) (world_t *world, buffer_t *game_log)
Add a resource in the given world and add the according command in the game_log buffer.

2.37.1 Function Documentation

2.37.1.1 `init_random_pos()`

```
void init_random_pos (
    vec2i_t * pos,
    int width,
    int height )
```

Fill a vector with random value, boundaries are width and height.

Parameters

<i>pos</i>	
<i>width</i>	
<i>height</i>	

Definition at line 29 of file `spawn_food.c`.

2.37.1.2 `spawn_resource()`

```
void spawn_resource (
    world_t * world,
    buffer_t * game_log )
```

Add a resource in the given world and add the according command in the `game_log` buffer.

Parameters

<i>world</i>	
<i>game_log</i>	

Definition at line 38 of file `spawn_food.c`.

2.37.1.3 `spawn_resource_at_pos()`

```
bool spawn_resource_at_pos (
    world_t * map,
    vec2i_t * pos,
    resource_t resource )
```

Spawn a resource at the given pos in the world.

Parameters

<i>map</i>	
<i>pos</i>	
<i>resource</i>	

Definition at line 16 of file spawn_food.c.

2.38 server/game/player/add.c File Reference

```
#include "game/game.h"
#include "game/player.h"
#include "game/log.h"
```

Functions

- bool [init_player_tile](#) (player_t *player, world_t *world, bool random_pos, vec2i_t *pos)
Give a tile to the given player, could be random or via a position.
- player_t * [create_player](#) (player_info_t info, world_t *world, double f)
Allocate and fill a player structure with given info.
- player_t * [make_new_player](#) (game_t *game, player_info_t info)
Allocate, fill and add a player to the world.

2.38.1 Function Documentation

2.38.1.1 create_player()

```
player_t* create_player (
    player_info_t info,
    world_t * world,
    double f )
```

Allocate and fill a player structure with given info.

Parameters

<i>info</i>	
<i>world</i>	
<i>f</i>	

Returns

a pointer to allocated player

Definition at line 37 of file add.c.

2.38.1.2 init_player_tile()

```
bool init_player_tile (
    player_t * player,
    world_t * world,
    bool random_pos,
    vec2i_t * pos )
```

Give a tile to the given player, could be random or via a position.

Parameters

<i>player</i>	
<i>world</i>	
<i>random_pos</i>	
<i>pos</i>	

Returns

true or false, depending on success

Definition at line 18 of file add.c.

2.38.1.3 make_new_player()

```
player_t* make_new_player (
    game_t * game,
    player_info_t info )
```

Allocate, fill and add a player to the world.

Parameters

<i>game</i>	
<i>info</i>	

Returns

a pointer to the player

Definition at line 62 of file add.c.

2.39 server/network/client/add.c File Reference

```
#include <stdio.h>
#include "network/client.h"
```

Functions

- bool [append_to_client_list](#) (client_t **first, client_t *client)
Push a client into the given client linked list.

2.39.1 Function Documentation

2.39.1.1 [append_to_client_list\(\)](#)

```
bool append_to_client_list (
    client_t ** first,
    client_t * client )
```

Push a client into the given client linked list.

Parameters

<i>first</i>	
<i>client</i>	

Returns

true if success, false otherwise

Definition at line 15 of file add.c.

2.40 server/game/player/append.c File Reference

```
#include "game/player.h"
```

Functions

- bool [append_to_global_player_list](#) (player_t **first, player_t *player)
Push the given player to the given linked list.

2.40.1 Function Documentation

2.40.1.1 `append_to_global_player_list()`

```
bool append_to_global_player_list (
    player_t ** first,
    player_t * player )
```

Push the given player to the given linked list.

Parameters

<i>first</i>	
<i>player</i>	

Returns

true or false, depending on success

Definition at line 14 of file `append.c`.

2.41 server/game/player/check.c File Reference

```
#include "game/player.h"
```

Functions

- bool [has_pending_commands](#) (player_t *player)
check if the player have commands in his buffer to be executed

2.41.1 Function Documentation

2.41.1.1 `has_pending_commands()`

```
bool has_pending_commands (
    player_t * player )
```

check if the player have commands in his buffer to be executed

Parameters

<i>player</i>	
---------------	--

Returns

true if commands, false if not

Definition at line 13 of file check.c.

2.42 server/game/player/delete.c File Reference

```
#include "game/player.h"
```

Functions

- bool [remove_player_from_global_player_list](#) (player_t *player, player_t **list)
Remove the given player from the given linked list.
- void [free_player](#) (player_t *player)
Free the memory allocated for the given player.
- void [free_all_players](#) (player_t *player)
Delete a player from the game (remove and free), send him a "dead".
- void [remove_player](#) (player_t **player, player_t **global_list)
Remove a player from its tile and team and remove it from global player linked list.
- void [delete_player](#) (player_t **player, player_t **global_list)
Call remove player, send a message about its death to clients then free the memory allocated for it.

2.42.1 Function Documentation**2.42.1.1 delete_player()**

```
void delete_player (
    player_t ** player,
    player_t ** global_list )
```

Call remove player, send a message about its death to clients then free the memory allocated for it.

Parameters

<i>player</i>	
<i>global_list</i>	

Definition at line 79 of file delete.c.

2.42.1.2 free_all_players()

```
void free_all_players (
    player_t * player )
```

Delete a player from the game (remove and free), send him a "dead".

Parameters

<i>player</i>	
<i>global_list</i>	

Definition at line 50 of file delete.c.

2.42.1.3 free_player()

```
void free_player (
    player_t * player )
```

Free the memory allocated for the given player.

Parameters

<i>player</i>	
---------------	--

Definition at line 38 of file delete.c.

2.42.1.4 remove_player()

```
void remove_player (
    player_t ** player,
    player_t ** global_list )
```

Remove a player from its tile and team and remove it from global player linked list.

Parameters

<i>player</i>	
<i>global_list</i>	

Definition at line 66 of file delete.c.

2.42.1.5 remove_player_from_global_player_list()

```
bool remove_player_from_global_player_list (
    player_t * player,
    player_t ** list )
```

Remove the given player from the given linked list.

Parameters

<i>player</i>	
<i>list</i>	

Returns

true if removed, false otherwise

Definition at line 15 of file delete.c.

2.43 server/network/client/delete.c File Reference

```
#include <stdio.h>
#include "network/client.h"
#include "network/server.h"
```

Functions

- void [free_client](#) (client_t *client)
Close the client file descriptor and free the memory allocated for the client.
- void [free_all_clients](#) (client_t *client)
- void [delete_client](#) (client_t **first, int fd)
Remove the client corresponding to the given file descriptor from the given clients linked list.
- void [disconnect_client](#) (client_t **client, server_t *server, const char *reason)
Remove a client from the fdsets (it won't be able to communicate) and delete it.

2.43.1 Function Documentation

2.43.1.1 delete_client()

```
void delete_client (
    client_t ** first,
    int fd )
```

Remove the client corresponding to the given file descriptor from the given clients linked list.

Parameters

<i>first</i>	
<i>fd</i>	

Definition at line 40 of file delete.c.

2.43.1.2 disconnect_client()

```
void disconnect_client (
    client_t ** client,
    server_t * server,
    const char * reason )
```

Remove a client from the fdsets (it won't be able to communicate) and delete it.

Parameters

<i>client</i>	
<i>server</i>	
<i>reason</i>	

Definition at line 67 of file delete.c.

2.43.1.3 free_all_clients()

```
void free_all_clients (
    client_t * client )
```

Definition at line 24 of file delete.c.

2.43.1.4 free_client()

```
void free_client (
    client_t * client )
```

Close the client file descriptor and free the memory allocated for the client.

Parameters

<i>client</i>	
---------------	--

Definition at line 15 of file delete.c.

2.44 server/game/player/get.c File Reference

```
#include "game/player.h"
```

Functions

- `player_t * get_last_global_player (player_t *first)`
Get the last player in the given's player team.
- `unsigned int get_nbr_players_global (player_t *first)`
Count how much players are ready.

2.44.1 Function Documentation

2.44.1.1 `get_last_global_player()`

```
player_t* get_last_global_player (  
    player_t * first )
```

Get the last player in the given's player team.

Parameters

<i>first</i>	
--------------	--

Returns

a pointer to the last team player

Definition at line 13 of file get.c.

2.44.1.2 `get_nbr_players_global()`

```
unsigned int get_nbr_players_global (  
    player_t * first )
```

Count how much players are ready.

Parameters

<i>first</i>	
--------------	--

Returns

the count

Definition at line 25 of file get.c.

2.45 server/network/client/get.c File Reference

```
#include "network/client.h"
```

Functions

- `client_t * get_client_by_fd (client_t *first, int fd)`
Find a client corresponding to the given file descriptor in a given client linked list.
- `client_t * get_last_client (client_t *first)`
Find the last item in a given client linked list.

2.45.1 Function Documentation**2.45.1.1 [get_client_by_fd\(\)](#)**

```
client_t* get_client_by_fd (
    client_t * first,
    int fd )
```

Find a client corresponding to the given file descriptor in a given client linked list.

Parameters

<i>first</i>	
<i>fd</i>	

Returns

a pointer to the found client

Definition at line 15 of file get.c.

2.45.1.2 [get_last_client\(\)](#)

```
client_t* get_last_client (
    client_t * first )
```

Find the last item in a given client linked list.

Parameters

<i>first</i>	
--------------	--

Returns

a pointer to the last client, NULL in case of error

Definition at line 25 of file get.c.

2.46 server/game/team/add_player.c File Reference

```
#include "game/team.h"
```

Functions

- bool [append_to_team_player_list](#) (team_t *team, player_t *player)
Add a player in a team.

2.46.1 Function Documentation

2.46.1.1 [append_to_team_player_list\(\)](#)

```
bool append_to_team_player_list (  
    team_t * team,  
    player_t * player )
```

Add a player in a team.

Parameters

<i>team</i>	
<i>player</i>	

Returns

true if successfully added, false otherwise

Definition at line 14 of file add_player.c.

2.47 server/game/team/get_team.c File Reference

```
#include <stdio.h>
#include "game/team.h"
```

Functions

- `team_t * get_team (team_t *teams, char *team_name)`
Get a team from a team name in the team linked list.

2.47.1 Function Documentation

2.47.1.1 `get_team()`

```
team_t* get_team (
    team_t * teams,
    char * team_name )
```

Get a team from a team name in the team linked list.

Parameters

<i>teams</i>	
<i>team_name</i>	

Returns

the found team, NULL otherwise

Definition at line 15 of file `get_team.c`.

2.48 server/game/team/remove_player.c File Reference

```
#include "game/team.h"
```

Functions

- `bool remove_player_from_team (player_t *player, team_t *team)`
Remove the player from the given team linked list.

2.48.1 Function Documentation

2.48.1.1 `remove_player_from_team()`

```
bool remove_player_from_team (
    player_t * player,
    team_t * team )
```

Remove the player from the given team linked list.

Parameters

<i>player</i>	
<i>team</i>	

Returns

true if removed, false if not

Definition at line 14 of file `remove_player.c`.

2.49 `server/game/update/check_if_game_won.c` File Reference

```
#include "game/team.h"
```

Functions

- bool [check_if_game_won](#) (team_t *teams)

2.49.1 Function Documentation

2.49.1.1 `check_if_game_won()`

```
bool check_if_game_won (
    team_t * teams )
```

Definition at line 10 of file `check_if_game_won.c`.

2.50 server/game/update/execute_player_commands.c File Reference

```
#include <stdio.h>
#include <ctype.h>
#include "macros.h"
#include "game/game.h"
#include "game/commands.h"
```

Functions

- int [strcmp_until_space](#) (char *s1, char *s2)
A custom strcmp that that stop at the next space it find.
- int [execute_player_command](#) (char *cmd, player_t *player, game_t *game)
Apply commands to the player.
- bool [fill_response_to_client](#) (c_buffer_t *buffer_response, int val)

2.50.1 Function Documentation

2.50.1.1 execute_player_command()

```
int execute_player_command (
    char * cmd,
    player_t * player,
    game_t * game )
```

Apply commands to the player.

Parameters

<i>cmd</i>	
<i>player</i>	
<i>game</i>	

Returns

Definition at line 38 of file execute_player_commands.c.

2.50.1.2 fill_response_to_client()

```
bool fill_response_to_client (
    c_buffer_t * buffer_response,
    int val )
```

Definition at line 53 of file execute_player_commands.c.

2.50.1.3 strcmp_until_space()

```
int strcmp_until_space (
    char * s1,
    char * s2 )
```

A custom strcmp that that stop at the next space it find.

Parameters

<i>s1</i>	
<i>s2</i>	

Returns

Definition at line 18 of file execute_player_commands.c.

2.51 server/game/update/set_player_command.c File Reference

```
#include "macros.h"
#include "game/commands.h"
#include "game/game.h"
#include "game/log.h"
```

Functions

- void [send_ritual_start_msg_to_players](#) (player_t *players)
Send a notification that an incantation has started to all clients.
- int [set_incantation_command](#) (player_t *player, buffer_t *game_log)
Check prerequisites and therefore call the actual incantation.
- int [set_command](#) (char *cmd, player_t *player, double f, buffer_t *game_log)
- bool [set_player_command](#) (player_t *player, double freq, buffer_t *game_log)

2.51.1 Function Documentation

2.51.1.1 send_ritual_start_msg_to_players()

```
void send_ritual_start_msg_to_players (
    player_t * players )
```

Send a notification that an incantation has started to all clients.

Parameters

<i>players</i>	
----------------	--

Definition at line 18 of file set_player_command.c.

2.51.1.2 set_command()

```
int set_command (
    char * cmd,
    player_t * player,
    double f,
    buffer_t * game_log )
```

Definition at line 54 of file set_player_command.c.

2.51.1.3 set_incantation_command()

```
int set_incantation_command (
    player_t * player,
    buffer_t * game_log )
```

Check prerequisites and therefore call the actual incantation.

Parameters

<i>player</i>	
<i>game_log</i>	

Returns

Definition at line 35 of file set_player_command.c.

2.51.1.4 set_player_command()

```
bool set_player_command (
    player_t * player,
    double freq,
    buffer_t * game_log )
```

Definition at line 70 of file set_player_command.c.

2.52 server/game/update/update_game.c File Reference

```
#include "zappy.h"
#include "game/log.h"
#include "macros.h"
```

Functions

- bool [handle_player_command](#) (player_t *player, game_t *game)
- bool [handle_players](#) (player_t *players, game_t *game)
Execute pending commands on players.
- bool [should_game_reset](#) (game_t *game)
- int [update_game](#) (game_t *game)
Update the game state.

2.52.1 Function Documentation

2.52.1.1 handle_player_command()

```
bool handle_player_command (
    player_t * player,
    game_t * game )
```

Definition at line 12 of file update_game.c.

2.52.1.2 handle_players()

```
bool handle_players (
    player_t * players,
    game_t * game )
```

Execute pending commands on players.

Parameters

<i>players</i>	
<i>game</i>	

Returns

Definition at line 33 of file update_game.c.

2.52.1.3 should_game_reset()

```
bool should_game_reset (
    game_t * game )
```

Definition at line 48 of file update_game.c.

2.52.1.4 update_game()

```
int update_game (
    game_t * game )
```

Update the game state.

Parameters

<i>game</i>	
-------------	--

Returns

Definition at line 64 of file update_game.c.

2.53 server/get_args/get_args.c File Reference

```
#include "arg_holder.h"
#include "zappy.h"
```

Functions

- void [free_arg_holder](#) (arg_holder_t *arg_hldr)
- int [check_arg_holder](#) (arg_holder_t *arg_hldr)
Check if all flags are filled.
- int [get_args_loop](#) (int ac, char **av, arg_holder_t *arg_hldr)
Loop through all cli args and redirect the value to the corresponding function.
- arg_holder_t * [get_args](#) (int ac, char **av)
Read args from the command line.

2.53.1 Function Documentation

2.53.1.1 check_arg_holder()

```
int check_arg_holder (
    arg_holder_t * arg_hldr )
```

Check if all flags are filled.

Parameters

<i>arg_hldr</i>	
-----------------	--

Returns

a status

Definition at line 20 of file get_args.c.

2.53.1.2 free_arg_holder()

```
void free_arg_holder (
    arg_holder_t * arg_hldr )
```

Definition at line 11 of file get_args.c.

2.53.1.3 get_args()

```
arg_holder_t* get_args (
    int ac,
    char ** av )
```

Read args from the command line.

Parameters

<i>ac</i>	
<i>av</i>	

Returns

Definition at line 70 of file get_args.c.

2.53.1.4 get_args_loop()

```
int get_args_loop (
    int ac,
    char ** av,
    arg_holder_t * arg_hldr )
```

Loop through all cli args and redirect the value to the corresponding function.

Parameters

<i>ac</i>	
<i>av</i>	
<i>arg_hldr</i>	

Returns

Definition at line 43 of file get_args.c.

2.54 server/get_args/options_func.c File Reference

```
#include "arg_holder.h"
#include "zappy.h"
```

Functions

- int [handle_opt_p](#) (char **arg, arg_holder_t *arg_holder, int *i)
- int [handle_opt_x](#) (char **arg, arg_holder_t *arg_holder, int *i)
- int [handle_opt_y](#) (char **arg, arg_holder_t *arg_holder, int *i)
- int [handle_opt_c](#) (char **arg, arg_holder_t *arg_holder, int *i)
- int [handle_opt_f](#) (char **arg, arg_holder_t *arg_holder, int *i)

2.54.1 Function Documentation

2.54.1.1 handle_opt_c()

```
int handle_opt_c (
    char ** arg,
    arg_holder_t * arg_holder,
    int * i )
```

Definition at line 68 of file options_func.c.

2.54.1.2 `handle_opt_f()`

```
int handle_opt_f (
    char ** arg,
    arg_holder_t * arg_holder,
    int * i )
```

Definition at line 87 of file `options_func.c`.

2.54.1.3 `handle_opt_p()`

```
int handle_opt_p (
    char ** arg,
    arg_holder_t * arg_holder,
    int * i )
```

Definition at line 11 of file `options_func.c`.

2.54.1.4 `handle_opt_x()`

```
int handle_opt_x (
    char ** arg,
    arg_holder_t * arg_holder,
    int * i )
```

Definition at line 30 of file `options_func.c`.

2.54.1.5 `handle_opt_y()`

```
int handle_opt_y (
    char ** arg,
    arg_holder_t * arg_holder,
    int * i )
```

Definition at line 49 of file `options_func.c`.

2.55 `server/get_args/options_names_func.c` File Reference

```
#include "arg_holder.h"
#include "zappy.h"
```


Functions

- bool [is_arg_option](#) (char *str)
- int [handle_opt_n](#) (char **arg, arg_holder_t *arg_holder, int *i)

2.55.1 Function Documentation

2.55.1.1 [handle_opt_n\(\)](#)

```
int handle_opt_n (  
    char ** arg,  
    arg_holder_t * arg_holder,  
    int * i )
```

Definition at line 16 of file options_names_func.c.

2.55.1.2 [is_arg_option\(\)](#)

```
bool is_arg_option (  
    char * str )
```

Definition at line 11 of file options_names_func.c.

2.56 server/init/game/game.c File Reference

```
#include "macros.h"  
#include "game/game.h"  
#include "arg_holder.h"
```

Functions

- int [init_game](#) (game_t *game, arg_holder_t *args)
Initialize the game (world, team, set the frequency)

2.56.1 Function Documentation

2.56.1.1 [init_game\(\)](#)

```
int init_game (  
    game_t * game,  
    arg_holder_t * args )
```

Initialize the game (world, team, set the frequency)

Parameters

<i>game</i>	
<i>args</i>	

Returns

Definition at line 16 of file game.c.

2.57 server/init/game/link_tiles.c File Reference

```
#include "game/map.h"  
#include "macros.h"
```

Functions

- void [link_tile](#) (int i, tile_t *init_tile, int width, int height)
Link a tile with his neighbours.
- void [link_tiles](#) (tile_t *map, int width, int height)
Link all tiles with each other, to make the map circular.

2.57.1 Function Documentation

2.57.1.1 link_tile()

```
void link_tile (  
    int i,  
    tile_t * init_tile,  
    int width,  
    int height )
```

Link a tile with his neighbours.

Parameters

<i>i</i>	
<i>init_tile</i>	
<i>width</i>	
<i>height</i>	

Definition at line 16 of file link_tiles.c.

2.57.1.2 link_tiles()

```
void link_tiles (
    tile_t * map,
    int width,
    int height )
```

Link all tiles with each other, to make the map circular.

Parameters

<i>map</i>	
<i>width</i>	
<i>height</i>	

Definition at line 42 of file link_tiles.c.

2.58 server/init/game/teams.c File Reference

```
#include "zappy.h"
#include "game/team.h"
```

Functions

- [team_t init_team](#) (char *name, int nb_clients_max)
Fill a team struct with the given name and max client.
- void [free_team_array](#) (team_t *teams)
Free the memory allocated by teams.
- team_t * [init_teams](#) (char **team_names, int nb_max_clients)
Allocate the memory required for teams and initialize them.
- void [print_teams](#) (team_t *teams)

2.58.1 Function Documentation

2.58.1.1 free_team_array()

```
void free_team_array (
    team_t * teams )
```

Free the memory allocated by teams.

Parameters

<i>teams</i>	
--------------	--

Definition at line 31 of file teams.c.

2.58.1.2 init_team()

```
team_t init_team (
    char * name,
    int nb_clients_max )
```

Fill a team struct with the given name and max client.

Parameters

<i>name</i>	
<i>nb_clients_max</i>	

Returns

Definition at line 15 of file teams.c.

2.58.1.3 init_teams()

```
team_t* init_teams (
    char ** team_names,
    int nb_max_clients )
```

Allocate the memory required for teams and initialize them.

Parameters

<i>team_names</i>	
<i>nb_max_clients</i>	

Returns

Definition at line 42 of file teams.c.

2.58.1.4 print_teams()

```
void print_teams (
    team_t * teams )
```

Definition at line 55 of file teams.c.

2.59 server/init/game/world.c File Reference

```
#include "game/map.h"
#include "errors.h"
#include "macros.h"
```

Functions

- void [free_map](#) (tile_t *map)
- int [init_world](#) (world_t *world, int width, int height)
Allocate the map, set its width and height and link all tiles together.

2.59.1 Function Documentation

2.59.1.1 free_map()

```
void free_map (
    tile_t * map )
```

Definition at line 12 of file world.c.

2.59.1.2 init_world()

```
int init_world (
    world_t * world,
    int width,
    int height )
```

Allocate the map, set its width and height and link all tiles together.

Parameters

<i>world</i>	
<i>width</i>	
<i>height</i>	

Returns

Definition at line 24 of file world.c.

2.60 server/init/server.c File Reference

```
#include "macros.h"
#include "errors.h"
#include "network/server.h"
```

Functions

- int [init_sock_address](#) (struct sockaddr_in *sock_addr, int socket, int port)
Bind the socket to the selected port.
- void [init_select](#) (server_t *server)
Setup select's fdsets.
- void [debug_no_reuse](#) (int sct)
- int [init_server](#) (server_t *server, int port)
Create the socket and setup the listening port.

2.60.1 Function Documentation

2.60.1.1 debug_no_reuse()

```
void debug_no_reuse (
    int sct )
```

Definition at line 40 of file server.c.

2.60.1.2 init_select()

```
void init_select (
    server_t * server )
```

Setup select's fdsets.

Parameters

<i>server</i>	
---------------	--

Definition at line 31 of file server.c.

2.60.1.3 init_server()

```
int init_server (
    server_t * server,
    int port )
```

Create the socket and setup the listening port.

Parameters

<i>server</i>	
<i>port</i>	

Returns

Definition at line 54 of file server.c.

2.60.1.4 init_sock_address()

```
int init_sock_address (
    struct sockaddr_in * sock_addr,
    int socket,
    int port )
```

Bind the socket to the selected port.

Parameters

<i>sock_addr</i>	
<i>socket</i>	
<i>port</i>	

Returns

Definition at line 17 of file server.c.

2.61 server/init/zappy.c File Reference

```
#include "zappy.h"
```

Functions

- void `free_zappy` (`zappy_t *zappy`)
- int `init_zappy` (`zappy_t *zappy`, `arg_holder_t *args`)
Global initialization function, call game and server ones.

2.61.1 Function Documentation

2.61.1.1 `free_zappy()`

```
void free_zappy (
    zappy_t * zappy )
```

Definition at line 10 of file `zappy.c`.

2.61.1.2 `init_zappy()`

```
int init_zappy (
    zappy_t * zappy,
    arg_holder_t * args )
```

Global initialization function, call game and server ones.

Parameters

<i>zappy</i>	
<i>args</i>	

Returns

Definition at line 28 of file `zappy.c`.

2.62 `server/main.c` File Reference

```
#include "zappy.h"
#include "game/map.h"
```

Functions

- int `zappy` (`arg_holder_t *args`)
Main game function, firstly init everything then launch the game and take care of freeing the memory after.
- int `main` (`int ac`, `char **av`)

2.62.1 Function Documentation

2.62.1.1 main()

```
int main (
    int ac,
    char ** av )
```

Definition at line 29 of file main.c.

2.62.1.2 zappy()

```
int zappy (
    arg_holder_t * args )
```

Main game function, firstly init everything then launch the game and take care of freeing the memory after.

Parameters

<i>args</i>	
-------------	--

Returns

ERR or OK

Definition at line 15 of file main.c.

2.63 server/network/client/accept.c File Reference

```
#include "network/server.h"
#include "macros.h"
#include "errors.h"
```

Functions

- int [get_socket_client](#) (int serv_socket, struct sockaddr_in *sock_addr)
- client_t * [accept_client](#) (server_t *server)
Accept (socket), create a client and add the client fd to the read and write fdsets.

2.63.1 Function Documentation

2.63.1.1 `accept_client()`

```
client_t* accept_client (
    server_t * server )
```

Accept (socket), create a client and add the client fd to the read and write fdsets.

Parameters

<code>server</code>	
---------------------	--

Returns

A pointer to the new client

Definition at line 34 of file `accept.c`.

2.63.1.2 `get_socket_client()`

```
int get_socket_client (
    int serv_socket,
    struct sockaddr_in * sock_addr )
```

Definition at line 12 of file `accept.c`.

2.64 `server/network/client/create.c` File Reference

```
#include "network/client.h"
#include "errors.h"
```

Functions

- `client_t* create_client` (int socket)
Allocate a client and setup base state.

2.64.1 Function Documentation

2.64.1.1 `create_client()`

```
client_t* create_client (
    int socket )
```

Allocate a client and setup base state.

Parameters

<code>socket</code>	
---------------------	--

Returns

a pointer to the allocated client

Definition at line 14 of file create.c.

2.65 server/network/client/reset_client_timer.c File Reference

```
#include "network/client.h"
```

Functions

- void [reset_client_timer](#) (client_t *client)

2.65.1 Function Documentation

2.65.1.1 reset_client_timer()

```
void reset_client_timer (  
    client_t * client )
```

Definition at line 10 of file reset_client_timer.c.

2.66 server/network/recv/recv_client_info.c File Reference

```
#include "network/server.h"
```

Functions

- int [recv_client_info](#) (client_t *client)

2.66.1 Function Documentation

2.66.1.1 `recv_client_info()`

```
int recv_client_info (
    client_t * client )
```

Definition at line 10 of file `recv_client_info.c`.

2.67 `server/network/rfds/give_player_to_client.c` File Reference

```
#include "zappy.h"
```

Functions

- `player_t * check_for_free_player` (`player_t *pls`, `team_t *team`, `client_ai_t *cli`)
Go over all players and check if some don't have a client connected on.
- `int give_player_to_client` (`client_ai_t *client`, `game_t *game`, `char *team_name`)
Attribute a player to a client, if none available, it will create a new one.

2.67.1 Function Documentation

2.67.1.1 `check_for_free_player()`

```
player_t* check_for_free_player (
    player_t * pls,
    team_t * team,
    client_ai_t * cli )
```

Go over all players and check if some don't have a client connected on.

Parameters

<i>pls</i>	
<i>team</i>	
<i>cli</i>	

Returns

Definition at line 16 of file `give_player_to_client.c`.

2.67.1.2 give_player_to_client()

```
int give_player_to_client (
    client_ai_t * client,
    game_t * game,
    char * team_name )
```

Attribute a player to a client, if none available, it will create a new one.

Parameters

<i>client</i>	
<i>game</i>	
<i>team_name</i>	

Returns

status

Definition at line 33 of file give_player_to_client.c.

2.68 server/network/rfds/handle_client.c File Reference

```
#include "game/log.h"
#include "zappy.h"
```

Functions

- bool [handle_client_ai](#) (client_ai_t **client, zappy_t *zappy)
Check if their command related to AI and execute them if the client is ready.
- bool [handle_client_graphic](#) (client_graphic_t **client, zappy_t *zappy)
- bool [give_player_to_client_unknown](#) (client_t **client, zappy_t *zappy, char *team_name)
Check if client is able to get a player, otherwise it will disconnect it.
- bool [handle_client_unknown](#) (client_t **client, zappy_t *zappy)
Determine if the client is a graphic client and update client structure to match it.
- bool [handle_client](#) (client_t **client, zappy_t *zappy)
Call the specific handle function according to the client type.

2.68.1 Function Documentation

2.68.1.1 give_player_to_client_unknown()

```
bool give_player_to_client_unknown (
    client_t ** client,
    zappy_t * zappy,
    char * team_name )
```

Check if client is able to get a player, otherwise it will disconnect it.

Parameters

<i>client</i>	
<i>zappy</i>	
<i>team_name</i>	

Returns

true on success, false otherwise

Definition at line 39 of file handle_client.c.

2.68.1.2 handle_client()

```
bool handle_client (
    client_t ** client,
    zappy_t * zappy )
```

Call the specific handle function according to the client type.

Parameters

<i>client</i>	
<i>zappy</i>	

Returns

Definition at line 93 of file handle_client.c.

2.68.1.3 handle_client_ai()

```
bool handle_client_ai (
    client_ai_t ** client,
    zappy_t * zappy )
```

Check if their command related to AI and execute them if the client is ready.

Parameters

<i>client</i>	
<i>zappy</i>	

Returns

Definition at line 16 of file handle_client.c.

2.68.1.4 handle_client_graphic()

```
bool handle_client_graphic (
    client_graphic_t ** client,
    zappy_t * zappy )
```

Definition at line 26 of file handle_client.c.

2.68.1.5 handle_client_unknown()

```
bool handle_client_unknown (
    client_t ** client,
    zappy_t * zappy )
```

Determine if the client is a graphic client and update client structure to match it.

Parameters

<i>client</i>	
<i>zappy</i>	

Returns

Definition at line 70 of file handle_client.c.

2.69 server/network/rfds/manage_rfds.c File Reference

```
#include "zappy.h"
#include "macros.h"
#include "network/server.h"
```

Functions

- bool [handle_new_connection](#) (zappy_t *zappy)
- int [manage_rfds](#) (client_t **client, zappy_t *zappy)

2.69.1 Function Documentation

2.69.1.1 `handle_new_connection()`

```
bool handle_new_connection (
    zappy_t * zappy )
```

Definition at line 12 of file `manage_rfds.c`.

2.69.1.2 `manage_rfds()`

```
int manage_rfds (
    client_t ** client,
    zappy_t * zappy )
```

Definition at line 23 of file `manage_rfds.c`.

2.70 `server/network/rfds/parse_command.c` File Reference

```
#include "network/client.h"
#include "macros.h"
#include <stdio.h>
```

Functions

- bool [remove_endline_from_command](#) (char *str)
- int [add_str_to_command](#) (c_buffer_t *buf, char **buf_str, char *cmd, int *j)
- bool [parse_command](#) (c_buffer_t *buf, char commands[MAX_COMMANDS][MAX_CMD_LEN])

2.70.1 Function Documentation

2.70.1.1 `add_str_to_command()`

```
int add_str_to_command (
    c_buffer_t * buf,
    char ** buf_str,
    char * cmd,
    int * j )
```

Definition at line 25 of file `parse_command.c`.

2.70.1.2 parse_command()

```
bool parse_command (
    c_buffer_t * buf,
    char commands[MAX_COMMANDS][MAX_CMD_LEN] )
```

Definition at line 46 of file parse_command.c.

2.70.1.3 remove_endline_from_command()

```
bool remove_endline_from_command (
    char * str )
```

Definition at line 12 of file parse_command.c.

2.71 server/network/select_server.c File Reference

```
#include <stdio.h>
#include "macros.h"
#include "zappy.h"
```

Functions

- bool [select_server](#) (zappy_t *zappy, fd_set *rfds, fd_set *wfds)
Allow the server to handle multiple clients concurrently.

2.71.1 Function Documentation

2.71.1.1 select_server()

```
bool select_server (
    zappy_t * zappy,
    fd_set * rfds,
    fd_set * wfds )
```

Allow the server to handle multiple clients concurrently.

Parameters

<i>zappy</i>	
<i>rfds</i>	
<i>wfds</i>	

Returns

true on success, false otherwise

Definition at line 17 of file select_server.c.

2.72 server/network/send/add_to_init_graph_response.c File Reference

```
#include "zappy.h"
```

Functions

- int [add_to_init_graph_response](#) (char *str, client_graphic_t *client)
Output a string to the buffer that'll be sent to the graphic client.

2.72.1 Function Documentation

2.72.1.1 add_to_init_graph_response()

```
int add_to_init_graph_response (  
    char * str,  
    client_graphic_t * client )
```

Output a string to the buffer that'll be sent to the graphic client.

Parameters

<i>str</i>	
<i>client</i>	

Returns

Definition at line 14 of file add_to_init_graph_response.c.

2.73 server/network/send/respond_to_client.c File Reference

```
#include "network/server.h"
```

Functions

- bool [respond_to_client](#) (client_t *client)
Send the content of the buffer to the client.

2.73.1 Function Documentation

2.73.1.1 respond_to_client()

```
bool respond_to_client (  
    client_t * client )
```

Send the content of the buffer to the client.

Parameters

<i>client</i>	
---------------	--

Returns

true if sent, false otherwise

Definition at line 13 of file respond_to_client.c.

2.74 server/network/send/send_info_to_graphical.c File Reference

```
#include "network/server.h"
```

Functions

- bool [send_info_to_graphical](#) (client_graphic_t *client, char *game_log)
Send commands contained in the game log to the graphic client.

2.74.1 Function Documentation

2.74.1.1 send_info_to_graphical()

```
bool send_info_to_graphical (  
    client_graphic_t * client,  
    char * game_log )
```

Send commands contained in the game log to the graphic client.

Parameters

<i>client</i>	
<i>game_log</i>	

Returns

true if sent, false otherwise

Definition at line 14 of file `send_info_to_graphical.c`.

2.75 server/network/send/send_init_info_to_graph.c File Reference

```
#include "zappy.h"
```

Functions

- bool [send_init_info_to_graph](#) (client_graphic_t *client, game_t *game)

2.75.1 Function Documentation**2.75.1.1 send_init_info_to_graph()**

```
bool send_init_info_to_graph (
    client_graphic_t * client,
    game_t * game )
```

Definition at line 99 of file `send_init_info_to_graph.c`.

2.76 server/network/wfds/manage_wfds.c File Reference

```
#include "zappy.h"
#include "network/server.h"
#include "macros.h"
```

Functions

- int [handle_client_wfds](#) (client_t **client, zappy_t *zappy)

Handle the while progress of a client, sent it welcome and map information on connection, and redirect after depending on his type.
- int [manage_wfds](#) (client_t **client, zappy_t *zappy)

Check if the handle_client_wfds returned an error, of if the client is a valid one.

2.76.1 Function Documentation

2.76.1.1 handle_client_wfds()

```
int handle_client_wfds (
    client_t ** client,
    zappy_t * zappy )
```

Handle the while progress of a client, sent it welcome and map information on connection, and redirect after depending on his type.

Parameters

<i>client</i>	
<i>zappy</i>	

Returns

status (ADD_FAIL, ERR, OK)

Definition at line 18 of file manage_wfds.c.

2.76.1.2 manage_wfds()

```
int manage_wfds (
    client_t ** client,
    zappy_t * zappy )
```

Check if the handle_client_wfds returned an error, of if the client is a valid one.

Parameters

<i>client</i>	
<i>zappy</i>	

Returns

Definition at line 40 of file manage_wfds.c.

2.77 server/network/wfds/manage_wfds_ai.c File Reference

```
#include "zappy.h"
#include "network/server.h"
```

```
#include "macros.h"
```

Functions

- void [init_serv_info_buffer](#) (client_ai_t *client, world_t *world)
Fill the client out buffer with map size and client ID.
- int [handle_client_ai_wfds](#) (client_t **client, zappy_t *zappy)
Send world info to the AI client.

2.77.1 Function Documentation

2.77.1.1 [handle_client_ai_wfds\(\)](#)

```
int handle_client_ai_wfds (
    client_t ** client,
    zappy_t * zappy )
```

Send world info to the AI client.

Parameters

<i>client</i>	
<i>zappy</i>	

Returns

Definition at line 35 of file `manage_wfds_ai.c`.

2.77.1.2 [init_serv_info_buffer\(\)](#)

```
void init_serv_info_buffer (
    client_ai_t * client,
    world_t * world )
```

Fill the client out buffer with map size and client ID.

Parameters

<i>client</i>	
<i>world</i>	

Definition at line 15 of file manage_wfds_ai.c.

2.78 server/network/wfds/manage_wfds_graph.c File Reference

```
#include "zappy.h"
#include "macros.h"
```

Functions

- int [handle_client_graph_wfds](#) (client_t **client, zappy_t *zappy)
Send the world info if the graphic need it, and send game logs otherwise.

2.78.1 Function Documentation

2.78.1.1 handle_client_graph_wfds()

```
int handle_client_graph_wfds (
    client_t ** client,
    zappy_t * zappy )
```

Send the world info if the graphic need it, and send game logs otherwise.

Parameters

<i>client</i>	
<i>zappy</i>	

Returns

Definition at line 16 of file manage_wfds_graph.c.

2.79 server/network/wfds/set_wfds.c File Reference

```
#include "network/server.h"
```

Functions

- void [if_set_wfds](#) (client_t *client, fd_set *perm_wfds, char *game_log)
- void [if_not_set_wfds](#) (client_t *client, fd_set *perm_wfds, char *game_log)
- void [set_wfds](#) (server_t *server, char *game_log)
Allow clients to get data from server by adding them in the write file descriptor set.

2.79.1 Function Documentation

2.79.1.1 if_not_set_wfds()

```
void if_not_set_wfds (
    client_t * client,
    fd_set * perm_wfds,
    char * game_log )
```

Definition at line 24 of file set_wfds.c.

2.79.1.2 if_set_wfds()

```
void if_set_wfds (
    client_t * client,
    fd_set * perm_wfds,
    char * game_log )
```

Definition at line 10 of file set_wfds.c.

2.79.1.3 set_wfds()

```
void set_wfds (
    server_t * server,
    char * game_log )
```

Allow clients to get data from server by adding them in the write file descriptor set.

Parameters

<i>server</i>	
<i>game_log</i>	

Definition at line 46 of file set_wfds.c.

2.80 server/run_zappy.c File Reference

```
#include "zappy.h"
```


Functions

- int [run_zappy](#) (zappy_t *zappy)
Main loop of the game.

2.80.1 Function Documentation

2.80.1.1 run_zappy()

```
int run_zappy (  
    zappy_t * zappy )
```

Main loop of the game.

Parameters

<i>zappy</i>	
--------------	--

Returns

ERR or GAME_WIN

Definition at line 13 of file run_zappy.c.

2.81 server/tools/array_tools.c File Reference

```
#include "zappy.h"
```

Functions

- unsigned int [get_array_size](#) (char **array)

2.81.1 Function Documentation

2.81.1.1 get_array_size()

```
unsigned int get_array_size (  
    char ** array )
```

Definition at line 10 of file array_tools.c.

2.82 server/tools/buffer.c File Reference

```
#include "zappy.h"
```

Functions

- void [reset_buffer](#) (buffer_t *buffer)
Remove all data from a buffer.
- bool [add_str_to_buffer](#) (char *str, buffer_t *buffer)
Append data into a buffer.

2.82.1 Function Documentation

2.82.1.1 [add_str_to_buffer\(\)](#)

```
bool add_str_to_buffer (  
    char * str,  
    buffer_t * buffer )
```

Append data into a buffer.

Parameters

<i>str</i>	
<i>buffer</i>	

Returns

true on success

Definition at line 22 of file buffer.c.

2.82.1.2 [reset_buffer\(\)](#)

```
void reset_buffer (  
    buffer_t * buffer )
```

Remove all data from a buffer.

Parameters

<i>buffer</i>	
---------------	--

Definition at line 12 of file buffer.c.

2.83 server/tools/error/int.c File Reference

```
#include "errors.h"
#include "macros.h"
```

Functions

- int [error](#) (const char *func)
- int [error_close](#) (const char *func, int fd)

2.83.1 Function Documentation

2.83.1.1 error()

```
int error (
    const char * func )
```

Definition at line 11 of file int.c.

2.83.1.2 error_close()

```
int error_close (
    const char * func,
    int fd )
```

Definition at line 17 of file int.c.

2.84 server/tools/error/print.c File Reference

```
#include "errors.h"
#include "macros.h"
#include "arg_holder.h"
```

Functions

- int [error_print](#) (const char *msg)
- void * [error_print_ptr](#) (const char *msg)

2.84.1 Function Documentation

2.84.1.1 `error_print()`

```
int error_print (
    const char * msg )
```

Definition at line 12 of file `print.c`.

2.84.1.2 `error_print_ptr()`

```
void* error_print_ptr (
    const char * msg )
```

Definition at line 18 of file `print.c`.

2.85 `server/tools/error/ptr.c` File Reference

```
#include "errors.h"
#include "macros.h"
```

Functions

- void * [error_ptr](#) (const char *func)
- void * [error_close_ptr](#) (const char *func, int fd)

2.85.1 Function Documentation

2.85.1.1 `error_close_ptr()`

```
void* error_close_ptr (
    const char * func,
    int fd )
```

Definition at line 17 of file `ptr.c`.

2.85.1.2 error_ptr()

```
void* error_ptr (
    const char * func )
```

Definition at line 11 of file ptr.c.

2.86 server/tools/usage.c File Reference

```
#include "zappy.h"
```

Functions

- void [usage](#) (char *bin_name)

2.86.1 Function Documentation

2.86.1.1 usage()

```
void usage (
    char * bin_name )
```

Definition at line 10 of file usage.c.

Index

accept.c
 accept_client, 75
 get_socket_client, 76
accept_client
 accept.c, 75
add_player.c
 append_to_team_player_list, 56
add_player_to_tile.c
 append_to_tile_player_list, 34
add_resource_to_inventory
 transfer_item.c, 26
add_str_to_buffer
 buffer.c, 92
add_str_to_command
 parse_command.c, 82
add_to_buffer
 add_to_buffer.c, 3
add_to_buffer.c
 add_to_buffer, 3
 can_fit_in_buffer, 4
 get_len_left, 4
add_to_init_graph_response
 add_to_init_graph_response.c, 84
add_to_init_graph_response.c
 add_to_init_graph_response, 84
append.c
 append_to_global_player_list, 49
append_to_client_list
 network/client/add.c, 48
append_to_global_player_list
 append.c, 49
append_to_team_player_list
 add_player.c, 56
append_to_tile_player_list
 add_player_to_tile.c, 34
array_tools.c
 get_array_size, 91

broadcast.c
 cmd_broadcast, 10
 convert_angle_to_num, 10
 get_angle, 11
 get_origin, 11
 get_traj, 11
buffer.c
 add_str_to_buffer, 92
 reset_buffer, 92

can_fit_in_buffer
 add_to_buffer.c, 4

check.c
 has_pending_commands, 49
check_arg_holder
 get_args.c, 63
check_for_free_player
 give_player_to_client.c, 78
check_for_timeouts
 check_for_timeouts.c, 7
check_for_timeouts.c
 check_for_timeouts, 7
check_if_game_won
 check_if_game_won.c, 58
check_if_game_won.c
 check_if_game_won, 58
check_if_ritual_req_met
 incantation.c, 16
check_ritual_resources
 check_ritual_resources.c, 25
check_ritual_resources.c
 check_ritual_resources, 25
clock_tick
 clock_tick.c, 8
clock_tick.c
 clock_tick, 8
 clock_tick_clients, 8
 clock_tick_player_ttl, 8
 clock_tick_players, 9
clock_tick_clients
 clock_tick.c, 8
clock_tick_player_ttl
 clock_tick.c, 8
clock_tick_players
 clock_tick.c, 9
cmd_broadcast
 broadcast.c, 10
cmd_connect_nbr
 connect_nbr.c, 13
cmd_eject
 eject.c, 13
cmd_fork
 fork.c, 15
cmd_forward
 move.c, 22
cmd_incantation
 incantation.c, 16
cmd_inventory
 inventory.c, 18
cmd_left
 move.c, 22

- cmd_look
 - look.c, 19
- cmd_right
 - move.c, 22
- cmd_set
 - take_set.c, 23
- cmd_take
 - take_set.c, 24
- command_list
 - command_list.c, 12
- command_list.c
 - command_list, 12
- connect_nbr.c
 - cmd_connect_nbr, 13
- convert_angle_to_num
 - broadcast.c, 10
- create.c
 - create_client, 76
- create_client
 - create.c, 76
- create_player
 - game/player/add.c, 46
- debug_no_reuse
 - server.c, 72
- del_resource_from_inventory
 - transfer_item.c, 27
- delete_client
 - network/client/delete.c, 52
- delete_player
 - game/player/delete.c, 50
- dir_log
 - move_log.c, 30
- dir_to_str
 - eject.c, 14
- disconnect_client
 - network/client/delete.c, 53
- eject.c
 - cmd_eject, 13
 - dir_to_str, 14
 - init_eject_msg, 14
- error
 - int.c, 93
- error_close
 - int.c, 93
- error_close_ptr
 - ptr.c, 94
- error_print
 - print.c, 94
- error_print_ptr
 - print.c, 94
- error_ptr
 - ptr.c, 94
- execute_player_command
 - execute_player_commands.c, 59
- execute_player_commands.c
 - execute_player_command, 59
 - fill_response_to_client, 59
 - strcmp_until_space, 60
- fill_resp_look
 - look.c, 19
- fill_resp_look_row
 - look.c, 20
- fill_response_to_client
 - execute_player_commands.c, 59
- fork.c
 - cmd_fork, 15
- forward_log
 - move_log.c, 31
- free_all_clients
 - network/client/delete.c, 53
- free_all_players
 - game/player/delete.c, 51
- free_arg_holder
 - get_args.c, 64
- free_client
 - network/client/delete.c, 53
- free_map
 - world.c, 71
- free_player
 - game/player/delete.c, 51
- free_team_array
 - teams.c, 69
- free_zappy
 - zappy.c, 74
- game.c
 - init_game, 67
- game/player/add.c
 - create_player, 46
 - init_player_tile, 47
 - make_new_player, 47
- game/player/delete.c
 - delete_player, 50
 - free_all_players, 51
 - free_player, 51
 - remove_player, 51
 - remove_player_from_global_player_list, 52
- game/player/get.c
 - get_last_global_player, 54
 - get_nbr_players_global, 54
- get_angle
 - broadcast.c, 11
- get_args
 - get_args.c, 64
- get_args.c
 - check_arg_holder, 63
 - free_arg_holder, 64
 - get_args, 64
 - get_args_loop, 64
- get_args_loop
 - get_args.c, 64
- get_array_size
 - array_tools.c, 91
- get_client_by_fd
 - network/client/get.c, 55

- get_dir.c
 - get_dir_to_left, [34](#)
 - get_dir_to_right, [35](#)
- get_dir_to_left
 - get_dir.c, [34](#)
- get_dir_to_right
 - get_dir.c, [35](#)
- get_last_client
 - network/client/get.c, [55](#)
- get_last_global_player
 - game/player/get.c, [54](#)
- get_last_team_player
 - team/get_player.c, [39](#)
- get_last_tile_player
 - map/get_player.c, [38](#)
- get_len_left
 - add_to_buffer.c, [4](#)
- get_nbr_players_global
 - game/player/get.c, [54](#)
- get_nbr_players_of_lvl_on_tile
 - map/get_nbr_players.c, [35](#)
- get_nbr_players_on_tile
 - map/get_nbr_players.c, [36](#)
- get_origin
 - broadcast.c, [11](#)
- get_random_resource
 - get_resource.c, [25](#)
- get_random_tile
 - get_tile.c, [39](#)
- get_resource.c
 - get_random_resource, [25](#)
 - get_resource_from_str, [25](#)
- get_resource_from_str
 - get_resource.c, [25](#)
- get_socket_client
 - accept.c, [76](#)
- get_team
 - get_team.c, [57](#)
- get_team.c
 - get_team, [57](#)
- get_team_nbr_players
 - team/get_nbr_players.c, [37](#)
- get_team_nbr_players_free
 - team/get_nbr_players.c, [37](#)
- get_team_nbr_players_not_free
 - team/get_nbr_players.c, [37](#)
- get_tile
 - get_tile.c, [40](#)
- get_tile.c
 - get_random_tile, [39](#)
 - get_tile, [40](#)
 - get_tile_in_dir, [40](#)
- get_tile_in_dir
 - get_tile.c, [40](#)
- get_traj
 - broadcast.c, [11](#)
- give_player_to_client
 - give_player_to_client.c, [78](#)
- give_player_to_client.c
 - check_for_free_player, [78](#)
 - give_player_to_client, [78](#)
- give_player_to_client_unknown
 - handle_client.c, [79](#)
- handle_client
 - handle_client.c, [80](#)
- handle_client.c
 - give_player_to_client_unknown, [79](#)
 - handle_client, [80](#)
 - handle_client_ai, [80](#)
 - handle_client_graphic, [81](#)
 - handle_client_unknown, [81](#)
- handle_client_ai
 - handle_client.c, [80](#)
- handle_client_ai_wfds
 - manage_wfds_ai.c, [88](#)
- handle_client_graph_wfds
 - manage_wfds_graph.c, [89](#)
- handle_client_graphic
 - handle_client.c, [81](#)
- handle_client_unknown
 - handle_client.c, [81](#)
- handle_client_wfds
 - manage_wfds.c, [87](#)
- handle_new_connection
 - manage_rfds.c, [82](#)
- handle_opt_c
 - options_func.c, [65](#)
- handle_opt_f
 - options_func.c, [65](#)
- handle_opt_n
 - options_names_func.c, [67](#)
- handle_opt_p
 - options_func.c, [66](#)
- handle_opt_x
 - options_func.c, [66](#)
- handle_opt_y
 - options_func.c, [66](#)
- handle_player_command
 - update_game.c, [62](#)
- handle_players
 - update_game.c, [62](#)
- has_command
 - has_command.c, [5](#)
- has_command.c
 - has_command, [5](#)
- has_pending_commands
 - check.c, [49](#)
- hatch_log
 - hatch_log.c, [28](#)
- hatch_log.c
 - hatch_log, [28](#)
- if_not_set_wfds
 - set_wfds.c, [90](#)
- if_set_wfds
 - set_wfds.c, [90](#)

- incantation.c
 - check_if_ritual_req_met, 16
 - cmd_incantation, 16
 - ritual_data, 17
 - send_ko_to_ritual_players, 16
 - upgrade_ritual_players, 17
- incantation_end_log
 - incantation_log.c, 29
- incantation_log.c
 - incantation_end_log, 29
 - incantation_start_log, 29
- incantation_start_log
 - incantation_log.c, 29
- init_buffer.c
 - init_c_buffer, 5
- init_c_buffer
 - init_buffer.c, 5
- init_eject_msg
 - eject.c, 14
- init_game
 - game.c, 67
- init_player_tile
 - game/player/add.c, 47
- init_random_pos
 - spawn_food.c, 45
- init_select
 - server.c, 72
- init_serv_info_buffer
 - manage_wfds_ai.c, 88
- init_server
 - server.c, 73
- init_sock_address
 - server.c, 73
- init_team
 - teams.c, 70
- init_teams
 - teams.c, 70
- init_world
 - world.c, 71
- init_zappy
 - zappy.c, 74
- int.c
 - error, 93
 - error_close, 93
- inventory.c
 - cmd_inventory, 18
- is_arg_option
 - options_names_func.c, 67
- is_buf_readable
 - read_from_buffer.c, 6
- kill_log
 - kill_log.c, 30
- kill_log.c
 - kill_log, 30
- link_tile
 - link_tiles.c, 68
- link_tiles
 - link_tiles.c, 69
- link_tiles.c
 - link_tile, 68
 - link_tiles, 69
- look.c
 - cmd_look, 19
 - fill_resp_look, 19
 - fill_resp_look_row, 20
- look_print
 - look_tile.c, 20
- look_print_player
 - look_tile.c, 20
- look_tile
 - look_tile.c, 21
- look_tile.c
 - look_print, 20
 - look_print_player, 20
 - look_tile, 21
- main
 - main.c, 75
- main.c
 - main, 75
 - zappy, 75
- make_new_player
 - game/player/add.c, 47
- manage_rfds
 - manage_rfds.c, 82
- manage_rfds.c
 - handle_new_connection, 82
 - manage_rfds, 82
- manage_wfds
 - manage_wfds.c, 87
- manage_wfds.c
 - handle_client_wfds, 87
 - manage_wfds, 87
- manage_wfds_ai.c
 - handle_client_ai_wfds, 88
 - init_serv_info_buffer, 88
- manage_wfds_graph.c
 - handle_client_graph_wfds, 89
- map/get_nbr_players.c
 - get_nbr_players_of_lvl_on_tile, 35
 - get_nbr_players_on_tile, 36
- map/get_player.c
 - get_last_tile_player, 38
- move.c
 - cmd_forward, 22
 - cmd_left, 22
 - cmd_right, 22
- move_log.c
 - dir_log, 30
 - forward_log, 31
- move_player_in_dir
 - move_player_to_tile.c, 41
- move_player_to_tile
 - move_player_to_tile.c, 41
- move_player_to_tile.c
 - move_player_in_dir, 41

- move_player_to_tile, [41](#)
- network/client/add.c
 - append_to_client_list, [48](#)
- network/client/delete.c
 - delete_client, [52](#)
 - disconnect_client, [53](#)
 - free_all_clients, [53](#)
 - free_client, [53](#)
- network/client/get.c
 - get_client_by_fd, [55](#)
 - get_last_client, [55](#)
- options_func.c
 - handle_opt_c, [65](#)
 - handle_opt_f, [65](#)
 - handle_opt_p, [66](#)
 - handle_opt_x, [66](#)
 - handle_opt_y, [66](#)
- options_names_func.c
 - handle_opt_n, [67](#)
 - is_arg_option, [67](#)
- parse_command
 - parse_command.c, [82](#)
- parse_command.c
 - add_str_to_command, [82](#)
 - parse_command, [82](#)
 - remove_endline_from_command, [83](#)
- print.c
 - error_print, [94](#)
 - error_print_ptr, [94](#)
- print_teams
 - teams.c, [70](#)
- ptr.c
 - error_close_ptr, [94](#)
 - error_ptr, [94](#)
- read_from_buffer
 - read_from_buffer.c, [6](#)
- read_from_buffer.c
 - is_buf_readable, [6](#)
 - read_from_buffer, [6](#)
- recv_client_info
 - recv_client_info.c, [77](#)
- recv_client_info.c
 - recv_client_info, [77](#)
- remove_endline_from_command
 - parse_command.c, [83](#)
- remove_player
 - game/player/delete.c, [51](#)
- remove_player.c
 - remove_player_from_team, [58](#)
- remove_player_from_global_player_list
 - game/player/delete.c, [52](#)
- remove_player_from_team
 - remove_player.c, [58](#)
- remove_player_from_tile
 - remove_player_from_tile.c, [43](#)
- remove_player_from_tile.c
 - remove_player_from_tile, [43](#)
- remove_resources_from_tile
 - remove_resources_from_tile.c, [44](#)
- remove_resources_from_tile.c
 - remove_resources_from_tile, [44](#)
- reset_buffer
 - buffer.c, [92](#)
- reset_client_timer
 - reset_client_timer.c, [77](#)
- reset_client_timer.c
 - reset_client_timer, [77](#)
- respond_to_client
 - respond_to_client.c, [85](#)
- respond_to_client.c
 - respond_to_client, [85](#)
- ritual_data
 - incantation.c, [17](#)
- run_zappy
 - run_zappy.c, [91](#)
- run_zappy.c
 - run_zappy, [91](#)
- select_server
 - select_server.c, [83](#)
- select_server.c
 - select_server, [83](#)
- send_info_to_graphical
 - send_info_to_graphical.c, [85](#)
- send_info_to_graphical.c
 - send_info_to_graphical, [85](#)
- send_init_info_to_graph
 - send_init_info_to_graph.c, [86](#)
- send_init_info_to_graph.c
 - send_init_info_to_graph, [86](#)
- send_ko_to_ritual_players
 - incantation.c, [16](#)
- send_ritual_start_msg_to_players
 - set_player_command.c, [60](#)
- server.c
 - debug_no_reuse, [72](#)
 - init_select, [72](#)
 - init_server, [73](#)
 - init_sock_address, [73](#)
- server/circular_buffer/add_to_buffer.c, [3](#)
- server/circular_buffer/has_command.c, [4](#)
- server/circular_buffer/init_buffer.c, [5](#)
- server/circular_buffer/read_from_buffer.c, [6](#)
- server/clock/check_for_timeouts.c, [7](#)
- server/clock/clock_tick.c, [7](#)
- server/game/commands/broadcast.c, [9](#)
- server/game/commands/command_list.c, [12](#)
- server/game/commands/connect_nbr.c, [12](#)
- server/game/commands/eject.c, [13](#)
- server/game/commands/fork.c, [14](#)
- server/game/commands/incantation.c, [15](#)
- server/game/commands/inventory.c, [18](#)
- server/game/commands/look.c, [18](#)
- server/game/commands/look_tile.c, [20](#)

- server/game/commands/move.c, 21
- server/game/commands/take_set.c, 23
- server/game/inventory/check_ritual_resources.c, 24
- server/game/inventory/get_resource.c, 25
- server/game/inventory/transfer_item.c, 26
- server/game/log/hatch_log.c, 28
- server/game/log/incantation_log.c, 28
- server/game/log/kill_log.c, 29
- server/game/log/move_log.c, 30
- server/game/log/spawn_log.c, 31
- server/game/log/take_log.c, 32
- server/game/map/add_player_to_tile.c, 33
- server/game/map/get_dir.c, 34
- server/game/map/get_nbr_players.c, 35
- server/game/map/get_player.c, 38
- server/game/map/get_tile.c, 39
- server/game/map/move_player_to_tile.c, 41
- server/game/map/remove_player_from_tile.c, 43
- server/game/map/remove_resources_from_tile.c, 44
- server/game/map/spawn_food.c, 44
- server/game/player/add.c, 46
- server/game/player/append.c, 48
- server/game/player/check.c, 49
- server/game/player/delete.c, 50
- server/game/player/get.c, 54
- server/game/team/add_player.c, 56
- server/game/team/get_nbr_players.c, 36
- server/game/team/get_player.c, 38
- server/game/team/get_team.c, 57
- server/game/team/remove_player.c, 57
- server/game/update/check_if_game_won.c, 58
- server/game/update/execute_player_commands.c, 59
- server/game/update/set_player_command.c, 60
- server/game/update/update_game.c, 62
- server/get_args/get_args.c, 63
- server/get_args/options_func.c, 65
- server/get_args/options_names_func.c, 66
- server/init/game/game.c, 67
- server/init/game/link_tiles.c, 68
- server/init/game/teams.c, 69
- server/init/game/world.c, 71
- server/init/server.c, 72
- server/init/zappy.c, 73
- server/main.c, 74
- server/network/client/accept.c, 75
- server/network/client/add.c, 48
- server/network/client/create.c, 76
- server/network/client/delete.c, 52
- server/network/client/get.c, 55
- server/network/client/reset_client_timer.c, 77
- server/network/recv/recv_client_info.c, 77
- server/network/rfds/give_player_to_client.c, 78
- server/network/rfds/handle_client.c, 79
- server/network/rfds/manage_rfds.c, 81
- server/network/rfds/parse_command.c, 82
- server/network/select_server.c, 83
- server/network/send/add_to_init_graph_response.c, 84
- server/network/send/respond_to_client.c, 84
- server/network/send/send_info_to_graphical.c, 85
- server/network/send/send_init_info_to_graph.c, 86
- server/network/wfds/manage_wfds.c, 86
- server/network/wfds/manage_wfds_ai.c, 87
- server/network/wfds/manage_wfds_graph.c, 89
- server/network/wfds/set_wfds.c, 89
- server/run_zappy.c, 90
- server/tools/array_tools.c, 91
- server/tools/buffer.c, 92
- server/tools/error/int.c, 93
- server/tools/error/print.c, 93
- server/tools/error/ptr.c, 94
- server/tools/usage.c, 95
- set_command
 - set_player_command.c, 61
- set_incantation_command
 - set_player_command.c, 61
- set_player_command
 - set_player_command.c, 61
- set_player_command.c
 - send_ritual_start_msg_to_players, 60
 - set_command, 61
 - set_incantation_command, 61
 - set_player_command, 61
- set_wfds
 - set_wfds.c, 90
- set_wfds.c
 - if_not_set_wfds, 90
 - if_set_wfds, 90
 - set_wfds, 90
- should_game_reset
 - update_game.c, 63
- spawn_food.c
 - init_random_pos, 45
 - spawn_resource, 45
 - spawn_resource_at_pos, 45
- spawn_log.c
 - spawn_player_log, 31
 - spawn_resource_log, 32
- spawn_player_log
 - spawn_log.c, 31
- spawn_resource
 - spawn_food.c, 45
- spawn_resource_at_pos
 - spawn_food.c, 45
- spawn_resource_log
 - spawn_log.c, 32
- strcmp_until_space
 - execute_player_commands.c, 60
- take_log
 - take_log.c, 33
- take_log.c
 - take_log, 33
 - take_log_inventory, 33
- take_log_inventory
 - take_log.c, 33
- take_set.c
 - cmd_set, 23

- cmd_take, [24](#)
- team/get_nbr_players.c
 - get_team_nbr_players, [37](#)
 - get_team_nbr_players_free, [37](#)
 - get_team_nbr_players_not_free, [37](#)
- team/get_player.c
 - get_last_team_player, [39](#)
- teams.c
 - free_team_array, [69](#)
 - init_team, [70](#)
 - init_teams, [70](#)
 - print_teams, [70](#)
- transfer_item
 - transfer_item.c, [27](#)
- transfer_item.c
 - add_resource_to_inventory, [26](#)
 - del_resource_from_inventory, [27](#)
 - transfer_item, [27](#)
- update_game
 - update_game.c, [63](#)
- update_game.c
 - handle_player_command, [62](#)
 - handle_players, [62](#)
 - should_game_reset, [63](#)
 - update_game, [63](#)
- upgrade_ritual_players
 - incantation.c, [17](#)
- usage
 - usage.c, [95](#)
- usage.c
 - usage, [95](#)
- world.c
 - free_map, [71](#)
 - init_world, [71](#)
- zappy
 - main.c, [75](#)
- zappy.c
 - free_zappy, [74](#)
 - init_zappy, [74](#)