

QILFAU
0.0

Generated by Doxygen 1.7.3

Mon Jun 6 2011 01:02:57

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	_field_dis_cap Struct Reference	5
3.2	_graph Struct Reference	6
3.3	_item_heap Struct Reference	7
3.4	_link Struct Reference	7
3.5	_link_list Struct Reference	8
3.6	_node Struct Reference	9
3.7	_path_item Struct Reference	10
3.8	column Struct Reference	11
3.9	combined_linear_congruential_gen Struct Reference	11
3.9.1	Detailed Description	11
3.10	config Struct Reference	12
3.10.1	Detailed Description	13
3.10.2	Field Documentation	13
3.10.2.1	runtime_state	13
3.11	csma_conf Struct Reference	14
3.12	csma_state Struct Reference	15
3.13	dense_matrix Struct Reference	16
3.14	empirical_params Struct Reference	16
3.14.1	Detailed Description	17
3.15	event Struct Reference	17
3.15.1	Detailed Description	19
3.16	event_info Struct Reference	19
3.16.1	Detailed Description	21
3.17	event_list Struct Reference	21
3.17.1	Detailed Description	22
3.18	fifo_queue Struct Reference	22
3.18.1	Detailed Description	24
3.19	garbage Struct Reference	25
3.20	heap Struct Reference	25
3.21	linear_congruential_gen Struct Reference	25
3.21.1	Detailed Description	26
3.22	linked_list Struct Reference	26

3.22.1	Detailed Description	27
3.23	linked_list_manager Struct Reference	27
3.23.1	Detailed Description	27
3.24	matrix Struct Reference	28
3.25	measures Struct Reference	29
3.25.1	Detailed Description	30
3.26	network Struct Reference	31
3.27	network_node_list Struct Reference	32
3.28	packet Struct Reference	32
3.28.1	Detailed Description	33
3.29	packet_info Struct Reference	33
3.29.1	Detailed Description	34
3.30	packet_list Struct Reference	34
3.30.1	Detailed Description	35
3.31	queue_config Struct Reference	36
3.31.1	Detailed Description	36
3.32	queue_type Struct Reference	36
3.32.1	Detailed Description	38
3.33	queue_type_list Struct Reference	38
3.33.1	Detailed Description	39
3.34	random_config Struct Reference	39
3.34.1	Detailed Description	40
3.35	random_distribution Struct Reference	40
3.35.1	Detailed Description	41
3.36	row Struct Reference	41
3.37	shortest_path_distance Struct Reference	42
3.38	sp_distance_list Struct Reference	43
3.39	sparse_matrix Struct Reference	44
3.40	statistical_number Struct Reference	44
3.40.1	Detailed Description	45
3.41	stop_config Struct Reference	45
3.41.1	Detailed Description	46
3.42	sys_state Struct Reference	46
3.42.1	Detailed Description	48
3.43	system_state_operations Struct Reference	48
3.43.1	Detailed Description	50
3.44	test_params Struct Reference	51
3.45	time Union Reference	51
3.45.1	Detailed Description	52
3.46	trash Struct Reference	52
3.47	uniform_params Struct Reference	52
3.47.1	Detailed Description	53
3.48	weibull_params Struct Reference	53
3.48.1	Detailed Description	53
3.49	yy_buffer_state Struct Reference	53
3.49.1	Field Documentation	54
3.49.1.1	yy_bs_column	54
3.49.1.2	yy_bs_lineno	54
3.50	yy_trans_info Struct Reference	54
3.51	yalloc Union Reference	55

3.52 YYLTYPE Struct Reference	55
3.53 YYSTYPE Union Reference	56
4 File Documentation	57
4.1 def.h File Reference	57
4.1.1 Detailed Description	57
4.1.2 Typedef Documentation	57
4.1.2.1 TIME	57
4.2 error.c File Reference	58
4.2.1 Detailed Description	58
4.2.2 Function Documentation	58
4.2.2.1 error	58
4.2.2.2 gc_malloc	58
4.2.2.3 trash_clean	59
4.2.2.4 trash_collect_garbage	59
4.3 error.h File Reference	60
4.3.1 Detailed Description	62
4.3.2 Define Documentation	62
4.3.2.1 check_null_pointer	62
4.3.2.2 iprint	62
4.3.2.3 try	63
4.3.3 Function Documentation	63
4.3.3.1 error	63
4.3.3.2 gc_malloc	63
4.3.3.3 trash_clean	64
4.3.3.4 trash_collect_garbage	64
4.4 irand/irand.h File Reference	65
4.4.1 Detailed Description	66
4.4.2 Function Documentation	66
4.4.2.1 irand_gen_bernoulli	66
4.4.2.2 irand_gen_erlang	66
4.4.2.3 irand_gen_exp	67
4.4.2.4 irand_gen_int_uniform	67
4.4.2.5 irand_gen_pareto	68
4.4.2.6 irand_gen_random	68
4.4.2.7 irand_gen_random_real	69
4.4.2.8 irand_gen_srandom	69
4.4.2.9 irand_gen_srandom_real	70
4.4.2.10 irand_gen_uniform	70
4.4.2.11 irand_init	70
4.4.2.12 irand_new_seed	71
4.4.2.13 irand_random_seed	71
4.5 irand/rnddist.c File Reference	72
4.5.1 Detailed Description	72
4.5.2 Function Documentation	73
4.5.2.1 _composition_gen_rnd	73
4.5.2.2 _convolution_gen_rnd	73
4.5.2.3 _inverse_bernoulli	74
4.5.2.4 _inverse_empirical	74
4.5.2.5 _inverse_exp	74

4.5.2.6	<code>_inverse_gen_rnd</code>	75
4.5.2.7	<code>_inverse_geometric</code>	75
4.5.2.8	<code>_inverse_int_uniform</code>	75
4.5.2.9	<code>_inverse_pareto</code>	76
4.5.2.10	<code>_inverse_uniform</code>	76
4.5.2.11	<code>_inverse_weibull</code>	76
4.5.2.12	<code>irand_gen_bernoulli</code>	76
4.5.2.13	<code>irand_gen_erlang</code>	77
4.5.2.14	<code>irand_gen_exp</code>	77
4.5.2.15	<code>irand_gen_int_uniform</code>	78
4.5.2.16	<code>irand_gen_pareto</code>	78
4.5.2.17	<code>irand_gen_uniform</code>	79
4.6	<code>irand/rndnum.c</code> File Reference	79
4.6.1	Detailed Description	80
4.6.2	Typedef Documentation	81
4.6.2.1	<code>COMBINED_LINEAR_GEN</code>	81
4.6.2.2	<code>LINEAR_LEHMER_GEN</code>	81
4.6.3	Function Documentation	81
4.6.3.1	<code>_combined_linear_gen_init</code>	81
4.6.3.2	<code>_combined_linear_gen_num</code>	81
4.6.3.3	<code>_combined_linear_gen_real</code>	82
4.6.3.4	<code>_linear_lehmer_gen_init</code>	82
4.6.3.5	<code>_linear_lehmer_gen_num</code>	83
4.6.3.6	<code>_linear_lehmer_gen_real</code>	83
4.6.3.7	<code>irand_gen_random</code>	83
4.6.3.8	<code>irand_gen_random_real</code>	84
4.6.3.9	<code>irand_gen_srandom</code>	84
4.6.3.10	<code>irand_gen_srandom_real</code>	85
4.6.3.11	<code>irand_init</code>	85
4.6.3.12	<code>irand_new_seed</code>	86
4.6.3.13	<code>irand_random_seed</code>	86
4.7	<code>list/heap.c</code> File Reference	87
4.7.1	Detailed Description	87
4.8	<code>list/heap.h</code> File Reference	87
4.8.1	Detailed Description	87
4.9	<code>list/linked_list.c</code> File Reference	88
4.9.1	Detailed Description	88
4.9.2	Function Documentation	88
4.9.2.1	<code>linked_list_get_first</code>	88
4.9.2.2	<code>linked_list_init</code>	89
4.9.2.3	<code>linked_list_insert</code>	89
4.9.2.4	<code>linked_list_man_alloc</code>	89
4.9.2.5	<code>linked_list_man_get_first</code>	90
4.9.2.6	<code>linked_list_man_get_free_entry</code>	90
4.9.2.7	<code>linked_list_man_init</code>	91
4.9.2.8	<code>linked_list_man_init_conf</code>	91
4.9.2.9	<code>linked_list_man_insert</code>	92
4.9.2.10	<code>linked_list_man_remove</code>	93
4.9.2.11	<code>linked_list_remove</code>	93
4.9.2.12	<code>print_list</code>	93

4.10	list/linked_list.h File Reference	94
4.10.1	Detailed Description	95
4.10.2	Define Documentation	95
4.10.2.1	container_of	95
4.10.3	Typedef Documentation	96
4.10.3.1	LINKED_LIST	96
4.10.3.2	LINKED_LIST_MAN	96
4.10.4	Function Documentation	96
4.10.4.1	linked_list_get_first	96
4.10.4.2	linked_list_init	96
4.10.4.3	linked_list_insert	96
4.10.4.4	linked_list_man_alloc	97
4.10.4.5	linked_list_man_get_first	97
4.10.4.6	linked_list_man_get_free_entry	98
4.10.4.7	linked_list_man_init	99
4.10.4.8	linked_list_man_init_conf	99
4.10.4.9	linked_list_man_insert	100
4.10.4.10	linked_list_man_remove	100
4.10.4.11	linked_list_remove	101
4.10.4.12	print_list	101
4.11	netlib.c File Reference	102
4.11.1	Detailed Description	102
4.12	netsim/conf/config.c File Reference	102
4.12.1	Detailed Description	103
4.12.2	Function Documentation	103
4.12.2.1	config_init	103
4.12.2.2	config_parse_file	103
4.12.2.3	config_random_conf	104
4.12.2.4	config_setup	104
4.13	netsim/conf/config.h File Reference	105
4.13.1	Detailed Description	106
4.13.2	Typedef Documentation	106
4.13.2.1	CONFIG	106
4.13.2.2	QUEUE_CONF	106
4.13.2.3	RANDOM_CONF	106
4.13.2.4	STOP_CONF	106
4.13.3	Function Documentation	106
4.13.3.1	config_init	106
4.13.3.2	config_parse_file	107
4.13.3.3	config_random_conf	107
4.13.3.4	config_setup	107
4.14	netsim/csma.c File Reference	108
4.14.1	Detailed Description	109
4.14.2	Function Documentation	109
4.14.2.1	csma_allow_continue	109
4.14.2.2	csma_generate_access	110
4.14.2.3	csma_generate_arrival	110
4.14.2.4	csma_generate_collision	111
4.14.2.5	csma_generate_end_service	111
4.14.2.6	csma_generate_event	112

4.14.2.7	csma_get_next_event	113
4.14.2.8	csma_process_access_event	113
4.14.2.9	csma_process_arrival	114
4.14.2.10	csma_process_collision	115
4.14.2.11	csma_process_end_service	115
4.14.2.12	csma_process_event	116
4.14.2.13	csma_remove_event	116
4.14.2.14	csma_state_init	117
4.15	netsim/csma.h File Reference	118
4.15.1	Detailed Description	118
4.15.2	Function Documentation	119
4.15.2.1	csma_state_init	119
4.16	netsim/event.c File Reference	119
4.16.1	Detailed Description	120
4.16.2	Function Documentation	120
4.16.2.1	event_init	120
4.16.2.2	event_list_get_first	121
4.16.2.3	event_list_init	121
4.16.2.4	event_list_insert_event	122
4.16.2.5	event_list_is_empty	122
4.16.2.6	event_list_new_event	123
4.16.2.7	event_list_remove_event	123
4.16.2.8	event_save	124
4.16.2.9	print_event_list	124
4.16.2.10	test_event_list_insert	124
4.17	netsim/event.h File Reference	125
4.17.1	Detailed Description	126
4.17.2	Define Documentation	126
4.17.2.1	swap_prev_event	126
4.17.3	Typedef Documentation	127
4.17.3.1	EVENT	127
4.17.3.2	EVENT_LIST	127
4.17.3.3	EVENTINFO	127
4.17.4	Function Documentation	127
4.17.4.1	event_init	127
4.17.4.2	event_list_get_first	128
4.17.4.3	event_list_init	128
4.17.4.4	event_list_insert_event	129
4.17.4.5	event_list_is_empty	129
4.17.4.6	event_list_new_event	129
4.17.4.7	event_list_remove_event	130
4.17.4.8	event_save	131
4.17.4.9	test_event_list_insert	131
4.18	netsim/netsim.c File Reference	131
4.18.1	Detailed Description	132
4.18.2	Function Documentation	132
4.18.2.1	event_setup	132
4.18.2.2	netsim_start	133
4.18.2.3	pisas_sched	133
4.19	netsim/netsim.h File Reference	134

4.19.1	Detailed Description	134
4.19.2	Function Documentation	134
4.19.2.1	event_setup	134
4.19.2.2	netsim_start	135
4.20	netsim/queues/fifo.c File Reference	136
4.20.1	Detailed Description	136
4.20.2	Function Documentation	136
4.20.2.1	fifo_init	136
4.20.2.2	fifo_setup	137
4.21	netsim/queues/fifo.h File Reference	137
4.21.1	Detailed Description	138
4.21.2	Typedef Documentation	138
4.21.2.1	FIFO_QINFO	138
4.21.3	Function Documentation	138
4.21.3.1	fifo_init	138
4.21.3.2	fifo_setup	139
4.22	netsim/queues/measures.c File Reference	139
4.22.1	Detailed Description	139
4.22.2	Define Documentation	140
4.22.2.1	print_statistical_value	140
4.22.3	Function Documentation	140
4.22.3.1	measurement_collect_data	140
4.22.3.2	measures_init	141
4.22.3.3	print_measurement	141
4.23	netsim/queues/measures.h File Reference	141
4.23.1	Detailed Description	142
4.23.2	Typedef Documentation	142
4.23.2.1	MEASURES	142
4.23.3	Function Documentation	142
4.23.3.1	measurement_collect_data	142
4.23.3.2	measures_init	143
4.23.3.3	print_measurement	143
4.24	netsim/queues/packet.c File Reference	143
4.24.1	Detailed Description	144
4.24.2	Function Documentation	144
4.24.2.1	measurement_self_collect_data	144
4.24.2.2	packet_init	144
4.24.2.3	packet_list_config	145
4.24.2.4	packet_list_get_first	145
4.24.2.5	packet_list_init	146
4.24.2.6	packet_list_insert_packet	146
4.24.2.7	packet_list_is_empty	147
4.24.2.8	packet_list_new_packet	147
4.24.2.9	packet_list_remove_packet	148
4.24.2.10	test_packet_list_new_packet	148
4.25	netsim/queues/packet.h File Reference	149
4.25.1	Detailed Description	150
4.25.2	Typedef Documentation	150
4.25.2.1	PACKET	150
4.25.2.2	PACKET_INFO	150

4.25.2.3	PACKET_LIST	150
4.25.3	Function Documentation	151
4.25.3.1	measurement_self_collect_data	151
4.25.3.2	packet_init	151
4.25.3.3	packet_list_config	152
4.25.3.4	packet_list_get_first	152
4.25.3.5	packet_list_init	152
4.25.3.6	packet_list_insert_packet	153
4.25.3.7	packet_list_is_empty	153
4.25.3.8	packet_list_new_packet	154
4.25.3.9	packet_list_remove_packet	154
4.25.3.10	test_packet_list_new_packet	155
4.26	netsim/queues/queue_man.c File Reference	155
4.26.1	Detailed Description	156
4.26.2	Function Documentation	156
4.26.2.1	queue_man_activate_type	156
4.26.2.2	queue_man_init	156
4.26.2.3	queue_man_register_new_type	157
4.26.2.4	queue_man_unregister_type	157
4.27	netsim/queues/queue_man.h File Reference	158
4.27.1	Detailed Description	158
4.27.2	Typedef Documentation	159
4.27.2.1	QUEUE_MAN	159
4.27.3	Function Documentation	159
4.27.3.1	queue_man_activate_type	159
4.27.3.2	queue_man_init	159
4.27.3.3	queue_man_register_new_type	160
4.27.3.4	queue_man_unregister_type	160
4.28	netsim/sys_aqueue.c File Reference	161
4.28.1	Detailed Description	161
4.28.2	Function Documentation	162
4.28.2.1	_allow_continue	162
4.28.2.2	_generate_arrival	162
4.28.2.3	_generate_end_service	163
4.28.2.4	_generate_event	164
4.28.2.5	_get_next_event	164
4.28.2.6	_packet_from_event	165
4.28.2.7	_process_arrival	165
4.28.2.8	_process_end_service	166
4.28.2.9	_process_event	166
4.28.2.10	_remove_event	167
4.28.2.11	sys_state_init	168
4.29	netsim/sys_aqueue.h File Reference	168
4.29.1	Detailed Description	169
4.29.2	Typedef Documentation	169
4.29.2.1	SYS_STATE	169
4.29.3	Function Documentation	169
4.29.3.1	sys_state_init	169
4.30	network.c File Reference	170
4.30.1	Detailed Description	171

4.31	network.h File Reference	171
4.31.1	Detailed Description	172
4.32	quantile.h File Reference	172
4.32.1	Detailed Description	172
4.32.2	Define Documentation	172
4.32.2.1	pvalue_chi_id	172
4.32.2.2	pvalue_normal_id	173
4.33	stat_num.c File Reference	173
4.33.1	Detailed Description	174
4.33.2	Function Documentation	174
4.33.2.1	stat_num_init	174
4.33.2.2	stat_num_new_sample	174
4.33.2.3	stat_num_new_time_sample	175
4.34	stat_num.h File Reference	175
4.34.1	Detailed Description	175
4.34.2	Typedef Documentation	176
4.34.2.1	STAT_NUM	176
4.34.3	Function Documentation	176
4.34.3.1	stat_num_init	176
4.34.3.2	stat_num_new_sample	176
4.34.3.3	stat_num_new_time_sample	176
4.35	tests/chisqr.c File Reference	177
4.35.1	Detailed Description	177
4.35.2	Function Documentation	177
4.35.2.1	test_chisqr	177

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

_field_dis_cap	5
_graph	6
_item_heap	7
_link	7
_link_list	8
_node	9
_path_item	10
column	11
combined_linear_congruential_gen	11
config	12
csma_conf	14
csma_state	15
dense_matrix	16
empirical_params (Parameters of Empirical Discrete distribution)	16
event	17
event_info	19
event_list	21
fifo_queue	22
garbage	25
heap	25
linear_congruential_gen	25
linked_list	26
linked_list_manager	27
matrix	28
measures	29
network	31
network_node_list	32
packet	32
packet_info	33

packet_list	34
queue_config	36
queue_type	36
queue_type_list	38
random_config	39
random_distribution (Random distribution structure (a framework))	40
row	41
shortest_path_distance	42
sp_distance_list	43
sparse_matrix	44
statistical_number	44
stop_config	45
sys_state	46
system_state_operations (Functions of a simulated system)	48
test_params	51
time	51
trash	52
uniform_params (Parameters of Uniform distribution)	52
weibull_params (Parameters of Weibull distribution)	53
yy_buffer_state	53
yy_trans_info	54
yyalloc	55
YYLTYPE	55
YYSTYPE	56

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

def.h	57
error.c	58
error.h	60
netlib.c	102
network.c	170
network.h	171
quantile.h	172
random.h	??
stat_num.c	173
stat_num.h	175
graph/fault.h	??
graph/graph.h	??
graph/main_graph.h	??
graph/main_rwa.h	??
graph/matrix.h	??
graph/print_structures.h	??
graph/routing.h	??
irand/irand.h	65
irand/rnddist.c	72
irand/rndnum.c	79
list/heap.c	87
list/heap.h	87
list/linked_list.c	88
list/linked_list.h	94
matrix/matrix.h	??
netsim/csma.c	108
netsim/csma.h	118
netsim/event.c	119
netsim/event.h	125

netsim/ netsim.c	131
netsim/ netsim.h	134
netsim/ sys_aqueue.c	161
netsim/ sys_aqueue.h	168
netsim/conf/ config.c	102
netsim/conf/ config.h	105
netsim/conf/ lexer.h	??
netsim/conf/ parser.h	??
netsim/queues/ fifo.c	136
netsim/queues/ fifo.h	137
netsim/queues/ measures.c	139
netsim/queues/ measures.h	141
netsim/queues/ packet.c	143
netsim/queues/ packet.h	149
netsim/queues/ queue_man.c	155
netsim/queues/ queue_man.h	158
polirand/ random.h	??
ranlib/ ranlib.h	??
tests/ chisqr.c	177

Chapter 3

Data Structure Documentation

3.1 `_field_dis_cap` Struct Reference

Data Fields

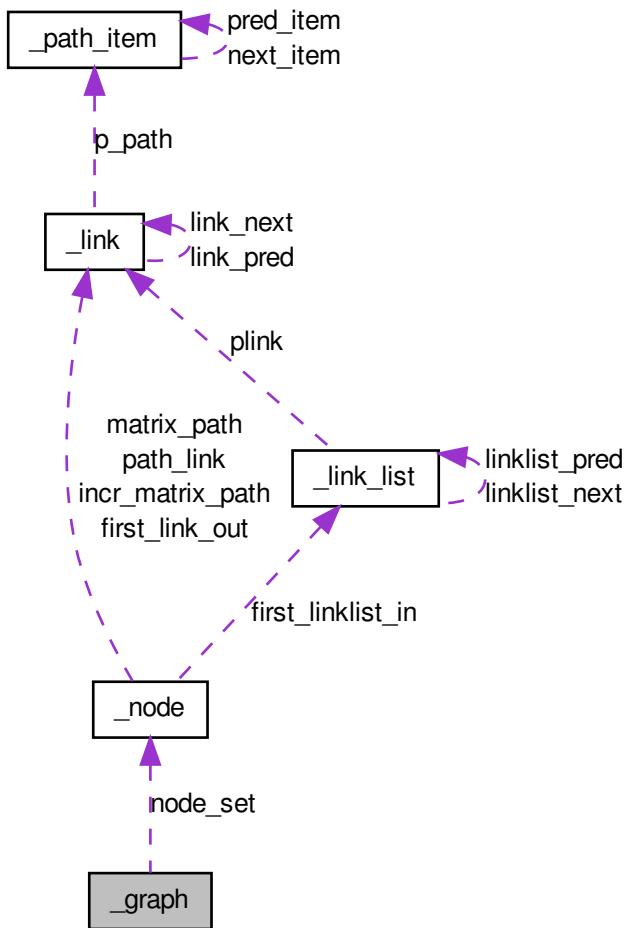
- double **dis_cap**
- double **cost_cap**

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

- graph/matrix.h

3.2 _graph Struct Reference

Collaboration diagram for _graph:



Data Fields

- int **max_num_nodes**
 - int **max_num_links**
 - int **num_nodes**
 - int **num_links**
 - COST **cost_graph**

- COST **cost_broken_graph**
- NODE * **node_set**

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

- graph/graph.h

3.3 _item_heap Struct Reference

Data Fields

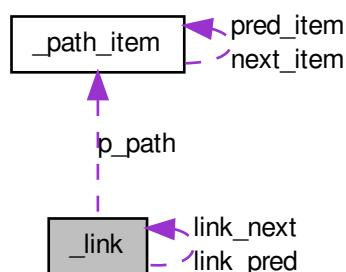
- NODEID **node_id**
- COST **path_cost**

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

- graph/routing.h

3.4 _link Struct Reference

Collaboration diagram for _link:



Data Fields

- int **identif**
- NODEID **node_from**
- NODEID **node_to**

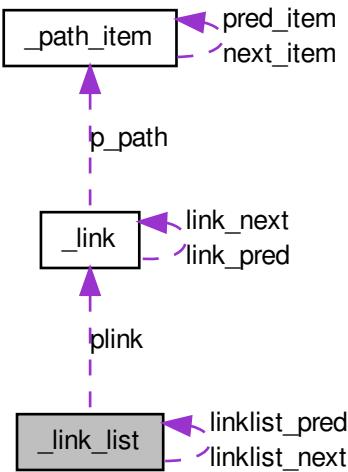
- COST(* **economic_cost**)(struct [_link](#) *p)
- COST **routing_cost**
- struct [_link](#) * **link_next**
- struct [_link](#) * **link_pred**
- BOOL **link_visited**
- BOOL **link_state**
- TRAFFIC **flow_aggr**
- TRAFFIC **flow_aggr_broken**
- TRAFFIC **max_flow_on_link**
- CAPACITY **capacity**
- CAPACITY **broken_capacity**
- [PATH_ITEM](#) * **p_path**

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

- graph/graph.h

3.5 [_link_list](#) Struct Reference

Collaboration diagram for [_link_list](#):



Data Fields

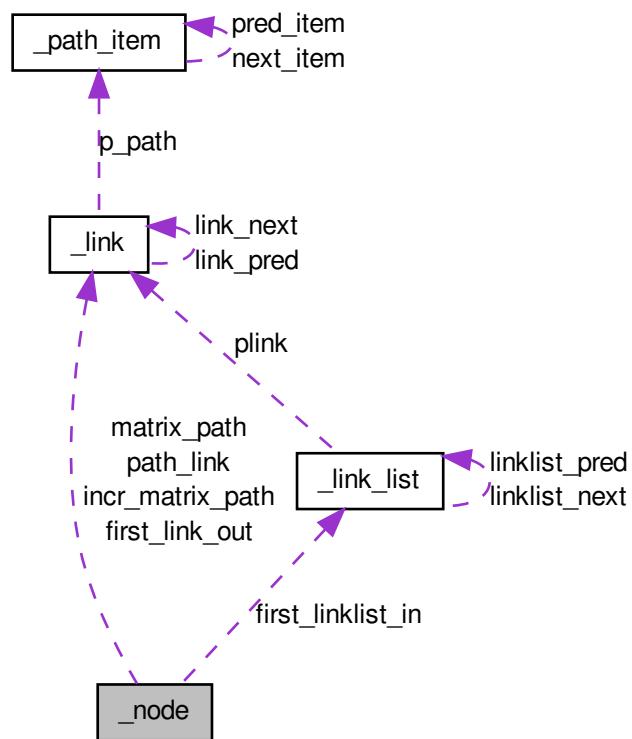
- **PLINK plink**
- struct [_link_list](#) * **linklist_next**
- struct [_link_list](#) * **linklist_pred**

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

- graph/graph.h

3.6 _node Struct Reference

Collaboration diagram for [_node](#):



Data Fields

- NODEID **node_id**
- **PLINK ** matrix_path**
- **PLINK ** incr_matrix_path**
- **PLINK first_link_out**
- **PLINKLIST first_linklist_in**
- int **num_links_out**
- BOOL **node_state**
- COST **path_weight**
- **PLINK path_link**

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

- graph/graph.h

3.7 _path_item Struct Reference

Collaboration diagram for _path_item:



Data Fields

- NODEID **node_from**
- NODEID **node_to**
- TRAFFIC **flow_item**
- struct **_path_item * next_item**
- struct **_path_item * pred_item**

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

- graph/graph.h

3.8 column Struct Reference

Data Fields

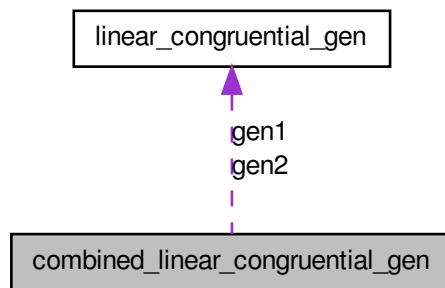
- int **id**
- float **data**

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

- matrix/matrix.h

3.9 combined_linear_congruential_gen Struct Reference

Collaboration diagram for combined_linear_congruential_gen:



Data Fields

- [LINEAR_LEHMER_GEN gen1](#)
The first Linear Congruential Generator.
- [LINEAR_LEHMER_GEN gen2](#)
The second Linear Congruential Generator.
- unsigned long **last_value**

3.9.1 Detailed Description

Random number generator based on Combined-Linear-Congruential Generator with two Linear Congruential Generator

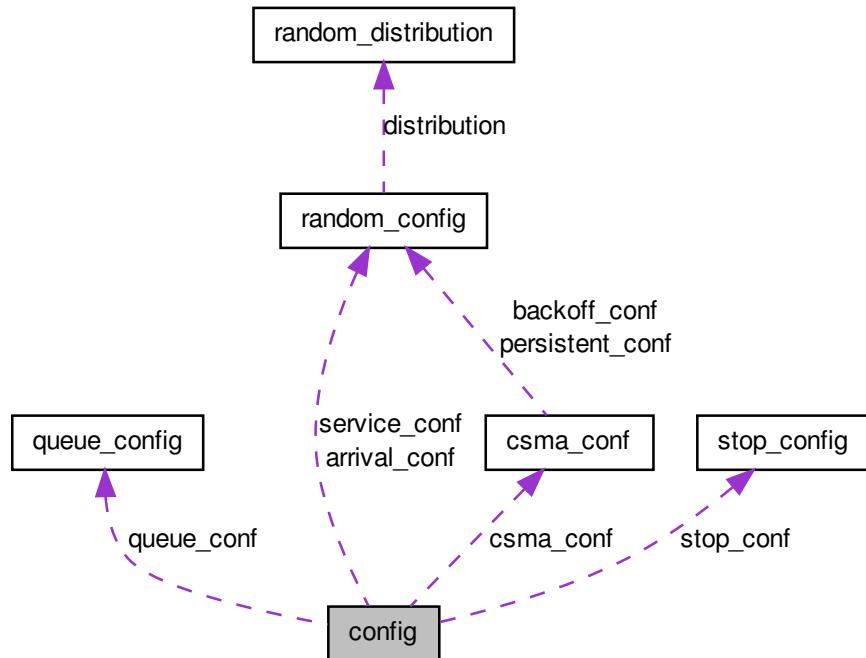
The documentation for this struct was generated from the following file:

- irand/rndnum.c

3.10 config Struct Reference

```
#include <config.h>
```

Collaboration diagram for config:



Data Fields

- `RANDOM_CONF arrival_conf`
Configuration of arrival flow.
- `RANDOM_CONF service_conf`
flow configuration of service time

- [QUEUE_CONF queue_conf](#)

Configuration of queue system.

- [STOP_CONF stop_conf](#)

Configuration of terminated conditions.

- int [random_lib](#)

Configuration of random library (IRAND, RANDLIB)

- [CSMA_CONF csma_conf](#)

CSMA configuration.

- int [protocol](#)

protocol: ONE_QUEUE or CSMA

- void * [runtime_state](#)

3.10.1 Detailed Description

Configuration from user

3.10.2 Field Documentation

3.10.2.1 void* config::runtime_state

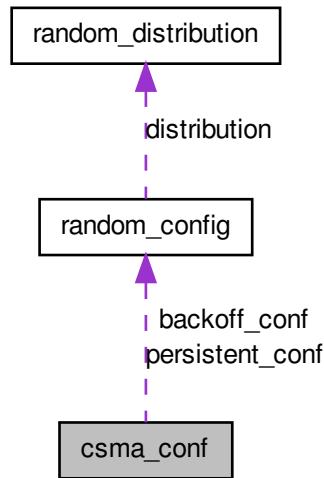
Current simulation: used for handling signal SIGINT (we dont want to wait to long time, but also want to see the intermediate result)

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

- netsim/conf/[config.h](#)

3.11 csma_conf Struct Reference

Collaboration diagram for csma_conf:



Data Fields

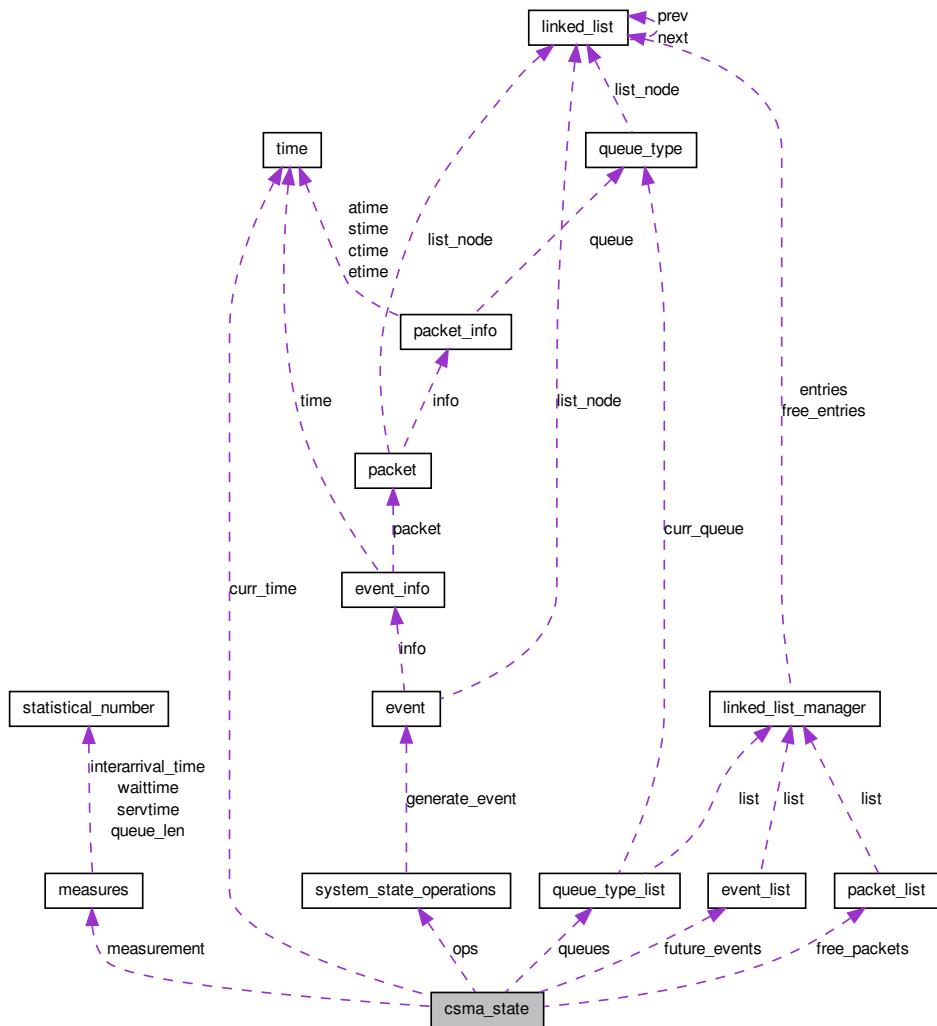
- double `slot_time`
slot time
- double `collision_time`
Collision time.
- int `nstations`
number of queues or station
- `RANDOM_CONF` `backoff_conf`
Backoff.
- `RANDOM_CONF` `persistent_conf`
persistent probability

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

- `netsim/conf/config.h`

3.12 csma_state Struct Reference

Collaboration diagram for csma_state:



Data Fields

- `SYS_STATE_OPS ops`

Abstract system operations.

- `int channel_state`

Channel state.

- **TIME curr_time**
Current time.
- **EVENT_LIST future_events**
List of future events (used for scheduling events)
- **QUEUE_MAN * queues**
Queue manager (support a list of queues)
- int **nqueues**
Number of queues.
- **MEASURES measurement**
Measurement information.
- **PACKET_LIST free_packets**
Free packet list (used to avoiding malloc operations).

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

- netsim/csma.h

3.13 dense_matrix Struct Reference

Data Fields

- float ** **vals**

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

- matrix/matrix.h

3.14 empirical_params Struct Reference

Parameters of Empirical Discrete distribution.

```
#include <irand.h>
```

Data Fields

- int [n](#)

Number of possible values.

- double * [values](#)

List of possible values.

- double * [probs](#)

List of probabilities for each values.

3.14.1 Detailed Description

Parameters of Empirical Discrete distribution.

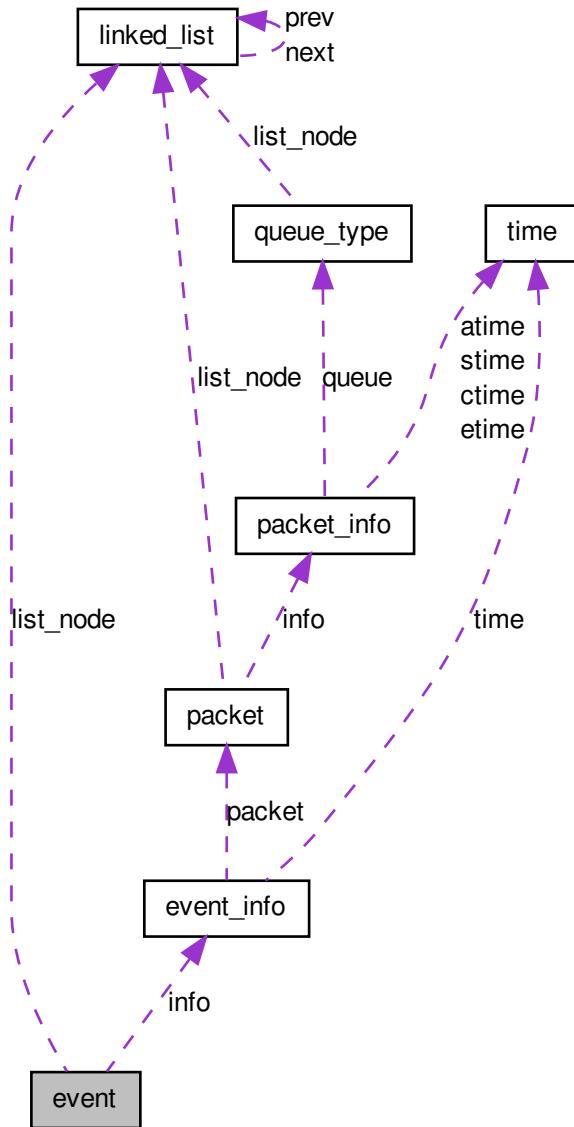
The documentation for this struct was generated from the following file:

- irand/[irand.h](#)

3.15 event Struct Reference

```
#include <event.h>
```

Collaboration diagram for event:



Data Fields

- [LINKED_LIST list_node](#)

Double linked list. This member is used to join in a list of event.

- [EVENTINFO info](#)

Event information.

3.15.1 Detailed Description

Event structure

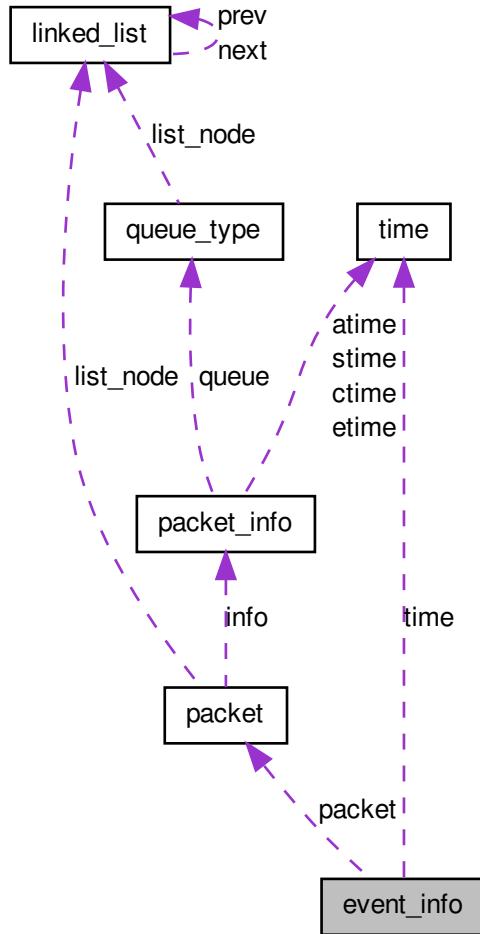
The documentation for this struct was generated from the following file:

- netsim/[event.h](#)

3.16 event_info Struct Reference

```
#include <event.h>
```

Collaboration diagram for event_info:



Data Fields

- `int type`
Type of event: EVENT_ARRIVAL, EVENT_END_SERVICE,...
- `TIME time`
Time that this event happens.
- `PACKET * packet`

Packet related to this event (used for end-service event)

3.16.1 Detailed Description

Information in an event

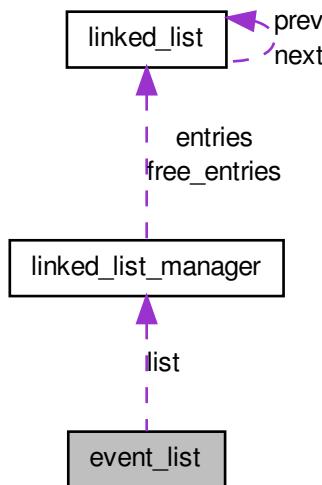
The documentation for this struct was generated from the following file:

- netsim/[event.h](#)

3.17 event_list Struct Reference

```
#include <event.h>
```

Collaboration diagram for event_list:



Data Fields

- [LINKED_LIST_MAN](#) `list`
Manager of double linked list of event.
- `int(* gen)(void *,...)`

3.17.1 Detailed Description

Event list structure

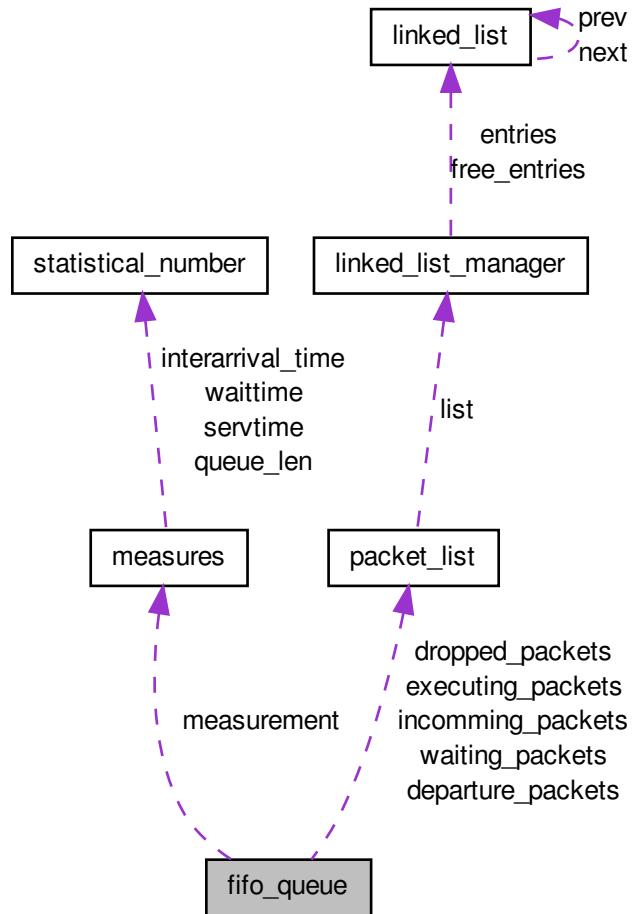
The documentation for this struct was generated from the following file:

- netsim/[event.h](#)

3.18 fifo_queue Struct Reference

```
#include <fifo.h>
```

Collaboration diagram for fifo_queue:



Data Fields

- **PACKET_LIST incomming_packets**
Incomming packet list.
- **PACKET_LIST waiting_packets**
Waiting packet list.
- **PACKET_LIST executing_packets**

Executing packet list.

- [PACKET_LIST dropped_packets](#)

Dropped packet list.

- [PACKET_LIST departure_packets](#)

Departure packet list.

- [MEASURES measurement](#)

Queue measurement.

- int [state](#)

Queue state.

- int [max_executing](#)

Maximum number of executing packets.

- int [max_waiting](#)

Maximum number of waiting packets.

3.18.1 Detailed Description

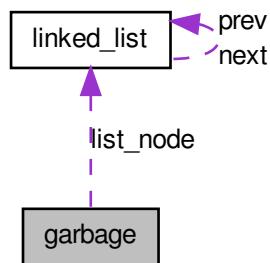
FIFO queue structure

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

- netsim/queues/[fifo.h](#)

3.19 garbage Struct Reference

Collaboration diagram for garbage:



Data Fields

- `LINKED_LIST list_node`
- `void * link`

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

- [error.h](#)

3.20 heap Struct Reference

Data Fields

- `int(* mapping)(double)`
- `void * entries`

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

- [list/heap.h](#)

3.21 linear_congruential_gen Struct Reference

Data Fields

- `unsigned long m`

Module.

- unsigned long **seed**
Seed or X0.
- unsigned long **mul**
Multiplier.
- unsigned long **inc**
Increment.
- unsigned long **last_value**
The last pseudo-random value.

3.21.1 Detailed Description

Linear Congruential Generator

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

- irand/rndnum.c

3.22 linked_list Struct Reference

```
#include <linked_list.h>
```

Collaboration diagram for linked_list:



Data Fields

- struct **linked_list** * **next**
Connect to previous node.
- struct **linked_list** * **prev**
Connect to next node.

3.22.1 Detailed Description

Linked list structure

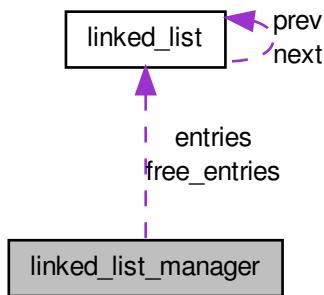
The documentation for this struct was generated from the following file:

- list/[linked_list.h](#)

3.23 linked_list_manager Struct Reference

```
#include <linked_list.h>
```

Collaboration diagram for linked_list_manager:



Data Fields

- [LINKED_LIST entries](#)
List of nodes.
- [LINKED_LIST free_entries](#)
List of free nodes.
- int [conf](#)
List configuration: storing entries or not, storing free nodes or not.

3.23.1 Detailed Description

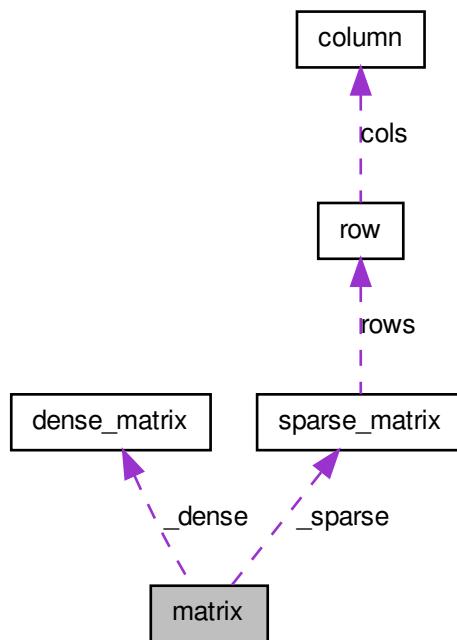
Linked list manager. Manage not only nodes of list but also free nodes (used for allocating new node)

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

- [list/linked_list.h](#)

3.24 matrix Struct Reference

Collaboration diagram for matrix:



Data Fields

- int **type**
- int **nrows**
- int **ncols**
- union {
 [DENSE_MATRIX](#) * **_dense**
 [SPARSE_MATRIX](#) * **_sparse**
} **data**

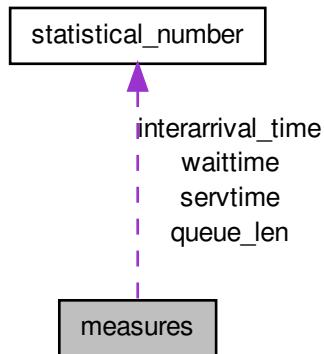
The documentation for this struct was generated from the following file:

- matrix/matrix.h

3.25 measures Struct Reference

```
#include <measures.h>
```

Collaboration diagram for measures:



Data Fields

- long **total_arrivals**
Total number of arrival events appearing in simulation.
- long **total_departures**
Total number of departure packets/events at output in simulation.
- long **total_dropped**
Total number of dropped packets.
- float **total_time**
Total time of simulation.
- STAT_NUM **queue_len**
Statistical value of queue length.

- [STAT_NUM servtime](#)

Statistical value of service time.

- [STAT_NUM waittime](#)

Statistical value of waiting time.

- [STAT_NUM interarrival_time](#)

Statistical value of inter-arrival time.

- float [last_arrival_time](#)

Last value of arrival time (temporary variable supporting to compute interarrival_time).

3.25.1 Detailed Description

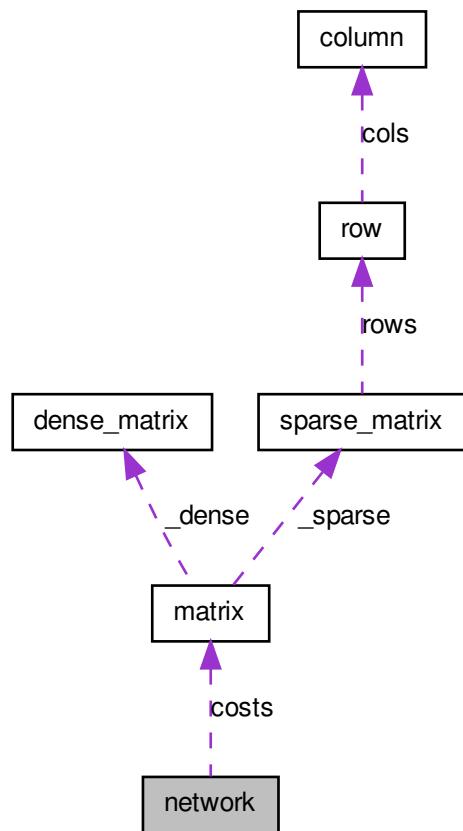
MEASURE structure used to store some results when simulating queue system

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

-
- netsim/queues/[measures.h](#)

3.26 network Struct Reference

Collaboration diagram for network:



Data Fields

- **MATRIX** `costs`

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

- `network.h`

3.27 network_node_list Struct Reference

Data Fields

- float * **data**
- int **nnodes**
- int **next_scan**

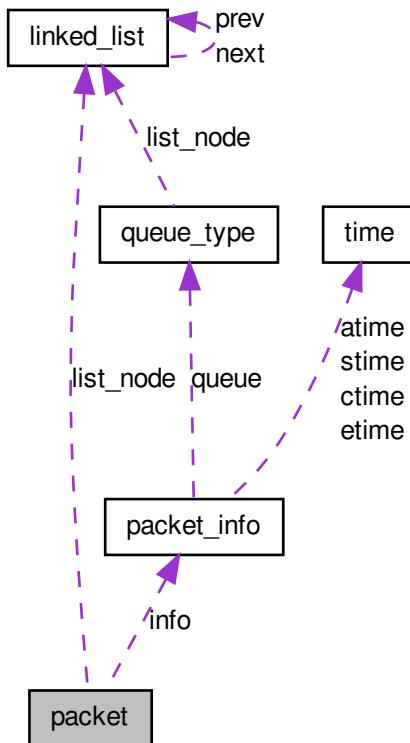
The documentation for this struct was generated from the following file:

- [network.h](#)

3.28 packet Struct Reference

```
#include <packet.h>
```

Collaboration diagram for packet:



Data Fields

- [LINKED_LIST list_node](#)
The element is used to connect to a packet-list.
- [PACKET_INFO info](#)
Packet information.

3.28.1 Detailed Description

Packet structure

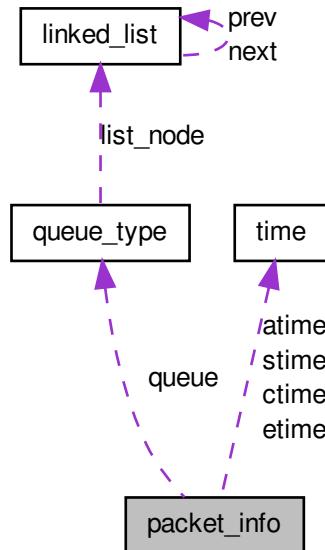
The documentation for this struct was generated from the following file:

- netsim/queues/[packet.h](#)

3.29 packet_info Struct Reference

```
#include <packet.h>
```

Collaboration diagram for packet_info:



Data Fields

- long `id`

Packet ID (currently no used)

- `QUEUE_TYPE * queue`

Queue.

- int `state`

State of packet: IN, WAITING, DROPPED, PROCESSING, OUT.

- float `service_time`

Service time of packet.

- `TIME atime`

Arrival time.

- `TIME ctime`

Collision time.

- `TIME stime`

Start time (time when packet is processed)

- `TIME etime`

End time (end of execution time)

3.29.1 Detailed Description

Packet information

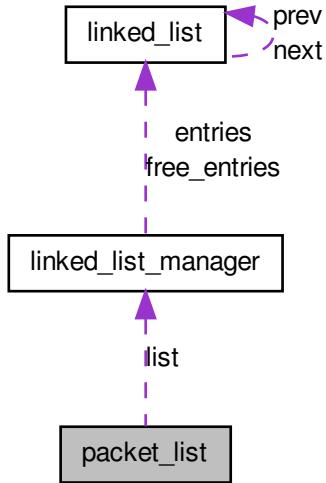
The documentation for this struct was generated from the following file:

- netsim/queues/`packet.h`

3.30 packet_list Struct Reference

```
#include <packet.h>
```

Collaboration diagram for packet_list:



Data Fields

- `LINKED_LIST_MAN list`
Manager of packet list.
- `int size`
Size of list.
- `int port_type`
Port type (no used now)
- `int port`
Port (no used now)

3.30.1 Detailed Description

Packet list structure

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

- [netsim/queues/packet.h](#)

3.31 queue_config Struct Reference

```
#include <config.h>
```

Data Fields

- int [type](#)

type of queue, now only support QUEUE_FIFO

- int [num_servers](#)

number of servers in a system

- int [max_waiters](#)

maximum number of clients allowing to wait in a system

- FILE * [out_file](#)

file name used to store event of departure flow

3.31.1 Detailed Description

Queue configuration

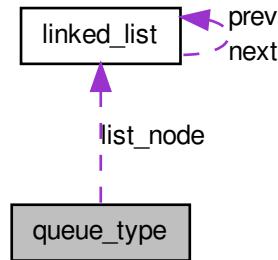
The documentation for this struct was generated from the following file:

- netsim/conf/[config.h](#)

3.32 queue_type Struct Reference

```
#include <queue_man.h>
```

Collaboration diagram for queue_type:



Data Fields

- `LINKED_LIST list_node`
This element is used to join to a queue-manager.
- `int type`
Type of queue. Now, only support QUEUE_FIFO.
- `int(* init)(QUEUE_TYPE *)`
Initialize queue type, include info in this structure.
- `int(* is_idle)(QUEUE_TYPE *)`
Check whether a queue is idle or not.
- `int(* push_packet)(QUEUE_TYPE *, PACKET *)`
Push a packet into queue.
- `int(* process_packet)(QUEUE_TYPE *, PACKET *)`
Process packet getting from queue.
- `int(* finish_packet)(QUEUE_TYPE *, PACKET *)`
Finish processing packet.
- `int(* get_waiting_length)(QUEUE_TYPE *)`
Get current queue length.
- `int(* select_waiting_packet)(QUEUE_TYPE *, PACKET **)`
Get a packet from waiting queue and remove it out of list.

- `int(* get_executing_packet)(QUEUE_TYPE *, PACKET **)`
Get a packet from executing queue.
- `int(* get_waiting_packet)(QUEUE_TYPE *, PACKET **)`
Get a packet from waiting queue, but packet still in queue.
- `void * info`

3.32.1 Detailed Description

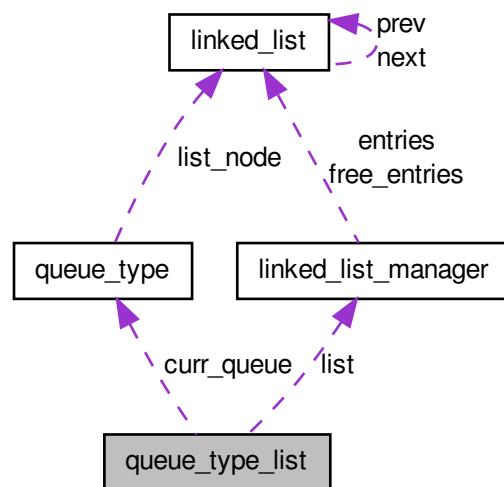
Structure of a queue type. Can be seen as an interface of a queue object
The documentation for this struct was generated from the following file:

- `netsim/queues/queue_man.h`

3.33 queue_type_list Struct Reference

```
#include <queue_man.h>
```

Collaboration diagram for queue_type_list:



Data Fields

- **LINKED_LIST_MAN** list
Manager of list of queue-type.
- **QUEUE_TYPE * curr_queue**
Current active queue_type.

3.33.1 Detailed Description

Queue manager structure

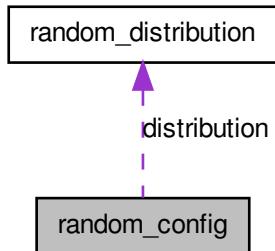
The documentation for this struct was generated from the following file:

- netsim/queues/[queue_man.h](#)

3.34 random_config Struct Reference

```
#include <config.h>
```

Collaboration diagram for random_config:

**Data Fields**

- int **type**
types of flow: RANDOM_MARKOVIAN, RANDOM_UNIFORM, RANDOM_FILE, RANDOM_OTHER.
- double **lambda**

used when type is RANDOM_MARKOVIAN

- double **from**
used for RANDOM_UNIFORM
- double **to**
used for RANDOM_UNIFORM
- double **prob**
used for FLOW_BERNOULLI
- FILE * **to_file**
file name that storing the this flow when it is generated
- FILE * **from_file**
used when type is RANDOM_FILE
- **RANDOM_DIST distribution**
Random Distribution of flow.

3.34.1 Detailed Description

Flow configuration is used to characterize a flow: what is its distribution, define some parameters.

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

- netsim/conf/[config.h](#)

3.35 random_distribution Struct Reference

Random distribution structure (a framework)

```
#include <irand.h>
```

Data Fields

- double(* **gen**)(**RANDOM_DIST** *)
Random number generator.
- double(* **cdf**)(**RANDOM_DIST** *, double)
Cumulative Distributed Function.
- void * **params**
Parameter of distribution.

3.35.1 Detailed Description

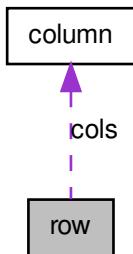
Random distribution structure (a framework)

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

- irand/irand.h

3.36 row Struct Reference

Collaboration diagram for row:



Data Fields

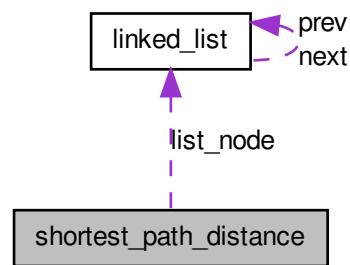
- int **id**
- int **ncols**
- [COLUMN * cols](#)

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

- matrix/matrix.h

3.37 shortest_path_distance Struct Reference

Collaboration diagram for shortest_path_distance:



Data Fields

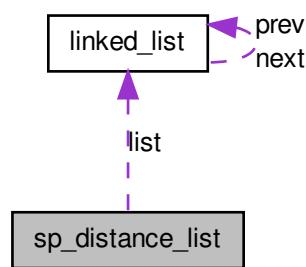
- [LINKED_LIST list_node](#)
- int **end_node**
- int **last_node**
- float **distance**

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

- [network.h](#)

3.38 sp_distance_list Struct Reference

Collaboration diagram for sp_distance_list:



Data Fields

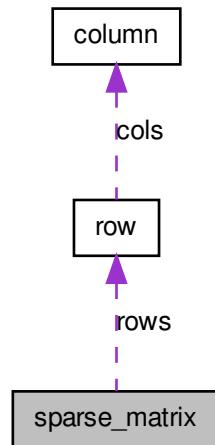
- `LINKED_LIST list`

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

- `network.h`

3.39 sparse_matrix Struct Reference

Collaboration diagram for sparse_matrix:



Data Fields

- int `nrows`
- `ROW * rows`

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

- matrix/matrix.h

3.40 statistical_number Struct Reference

```
#include <stat_num.h>
```

Data Fields

- float `min`
Minimum value.
- float `max`

Maximum value.

- float **avg**
Average value (mean)
- float **var**
Variance value.
- float **all_time**
Time of observing this random process.
- float **tmpsum**
Temporary value calculating sum of value.
- float **tmpsumsqr**
Temporary value calculating sum of value square.
- float **last_time**
The last time of observation.
- int **num_samples**
Number of samples in observation.

3.40.1 Detailed Description

Statistical information of a random process

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

- stat_num.h

3.41 stop_config Struct Reference

```
#include <config.h>
```

Data Fields

- float **max_time**
Maximum time is allowed to run simulation.
- int **max_arrival**
Maximum number of arrival events.
- int **queue_zero**
Stop when queue length is zero.

3.41.1 Detailed Description

Define conditions of stopping program

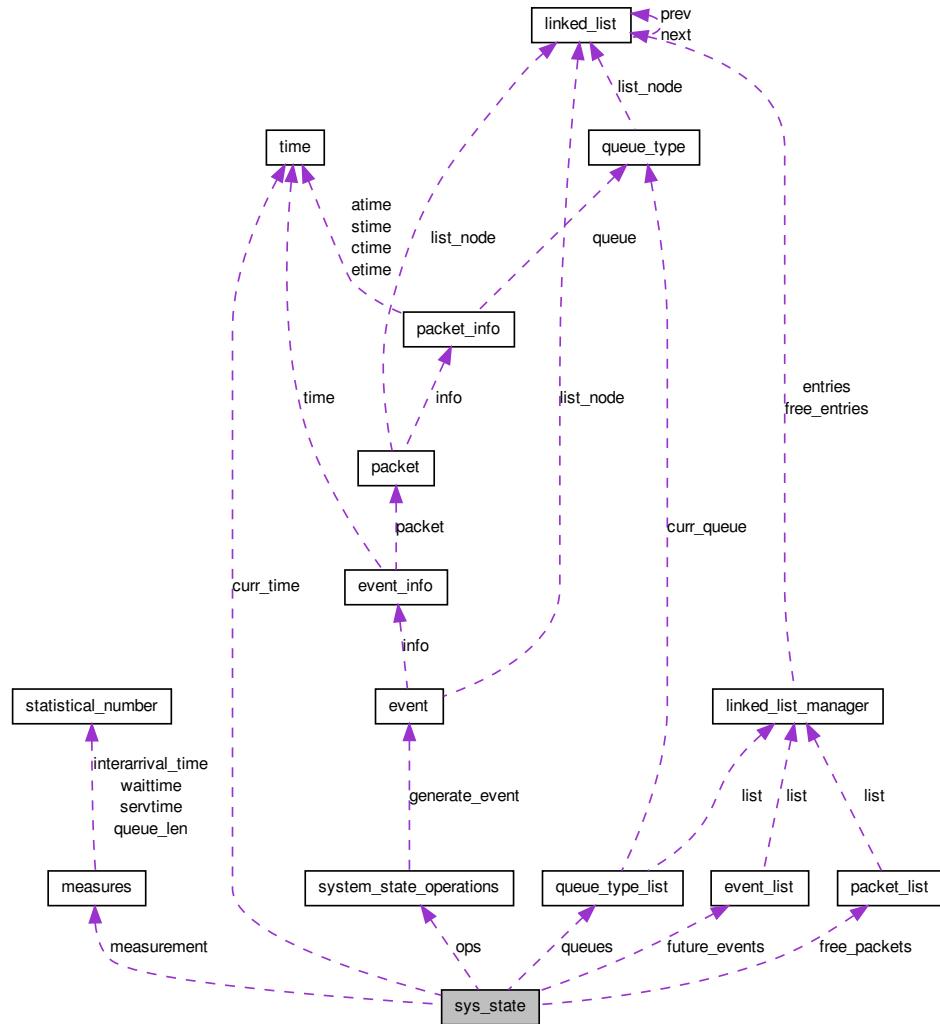
The documentation for this struct was generated from the following file:

- netsim/conf/[config.h](#)

3.42 sys_state Struct Reference

```
#include <sys_aqueue.h>
```

Collaboration diagram for sys_state:



Data Fields

- `SYS_STATE_OPS ops`

Abstract operations of a simulated system.

- TIME curr time

Current time.

- [EVENT_LIST future_events](#)

List of future events (used for scheduling events)

- [QUEUE_MAN queues](#)

Queue manager (support a list of queues)

- [MEASURES measurement](#)

Measurement information.

- [PACKET_LIST free_packets](#)

Free packet list (used to avoiding malloc operations).

3.42.1 Detailed Description

Structure representing the system state in simulation.

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

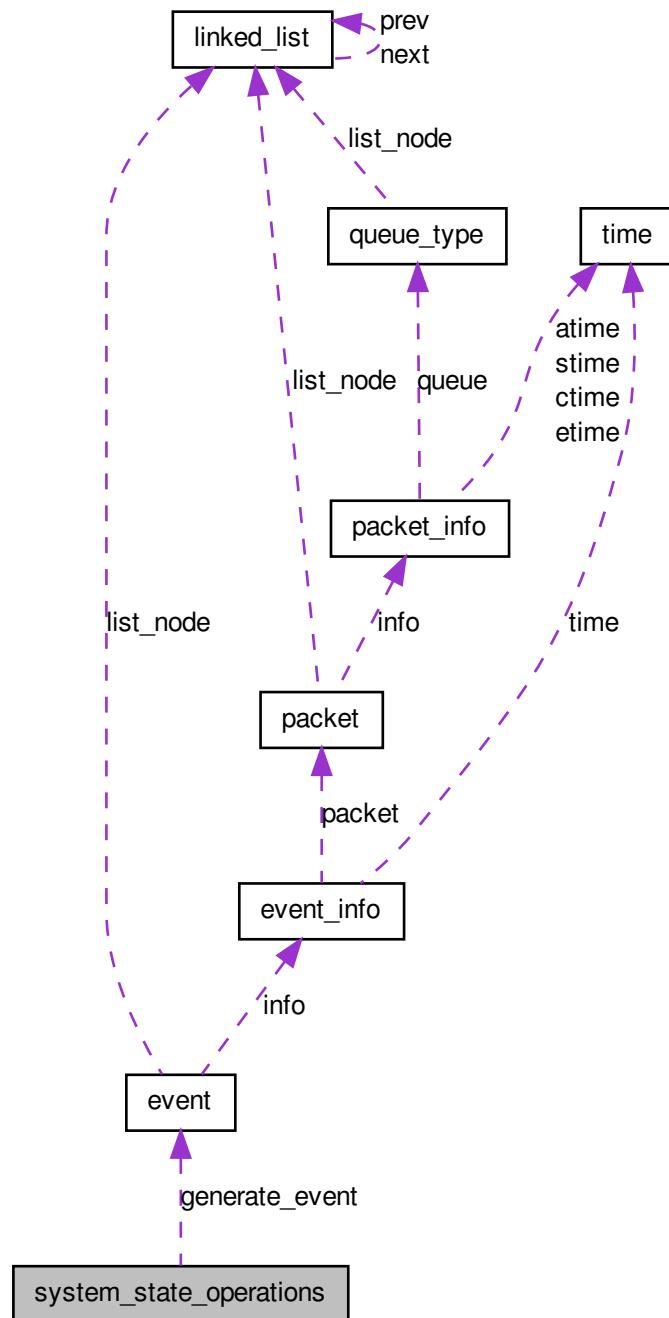
- netsim/[sys_aqueue.h](#)

3.43 system_state_operations Struct Reference

Functions of a simulated system.

```
#include <netsim.h>
```

Collaboration diagram for system_state_operations:



Data Fields

- `int(* get_next_event)(SYS_STATE_OPS *, EVENT **e)`

Get next event from event list.

- `int(* remove_event)(SYS_STATE_OPS *, EVENT *e)`

Remove an event out of event list.

- `int(* allow_continue)(CONFIG *, SYS_STATE_OPS *)`

Check whether the program is stopped (from user configuration)

- `EVENT *(* generate_event)(int type, PACKET *, CONFIG *, SYS_STATE_OPS *)`

Generate new event.

- `int(* process_event)(EVENT *e, CONFIG *, SYS_STATE_OPS *)`

Process an event.

- `int(* clean)(CONFIG *, SYS_STATE_OPS *)`

Clean the simulated system (when finishing simulation)

3.43.1 Detailed Description

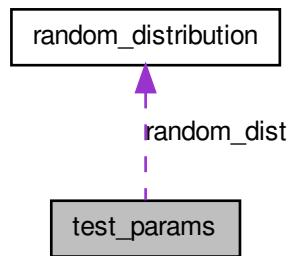
Functions of a simulated system.

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

- netsim/netsim.h

3.44 test_params Struct Reference

Collaboration diagram for test_params:



Data Fields

- int **nsamples**
- int **nintervals**
- double **p_value**
- RANDOM_DIST **random_dist**

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

- tests/[chisqr.c](#)

3.45 time Union Reference

```
#include <def.h>
```

Data Fields

- long **slot**
time is represented as slot time
- float **real**
time is considered as real time

3.45.1 Detailed Description

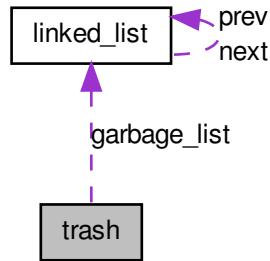
Time definition

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

- [def.h](#)

3.46 trash Struct Reference

Collaboration diagram for trash:



Data Fields

- [LINKED_LIST](#) `garbage_list`

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

- [error.h](#)

3.47 uniform_params Struct Reference

Parameters of Uniform distribution.

```
#include <irand.h>
```

Data Fields

- double `from`

Lower-bound value.

- double **to**

Upper-bound value.

3.47.1 Detailed Description

Parameters of Uniform distribution.

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

- irand/irand.h

3.48 weibull_params Struct Reference

Parameters of Weibull distribution.

```
#include <irand.h>
```

Data Fields

- double **a**

Value of a.

- double **b**

Value of b.

3.48.1 Detailed Description

Parameters of Weibull distribution.

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

- irand/irand.h

3.49 yy_buffer_state Struct Reference

Data Fields

- FILE * **yy_input_file**
- char * **yy_ch_buf**
- char * **yy_buf_pos**

- `yy_size_t yy_buf_size`
- `int yy_n_chars`
- `int yy_is_our_buffer`
- `int yy_is_interactive`
- `int yy_at_bol`
- `int yy_bs_lineno`
- `int yy_bs_column`
- `int yy_fill_buffer`
- `int yy_buffer_status`

3.49.1 Field Documentation

3.49.1.1 int yy_buffer_state::yy_bs_column

The column count.

3.49.1.2 int yy_buffer_state::yy_bs_lineno

The line count.

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

- netsim/conf/lexer.c
- netsim/conf/lexer.h

3.50 yy_trans_info Struct Reference

Data Fields

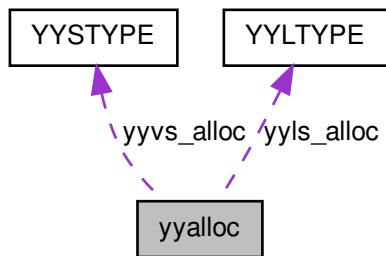
- `flex_int32_t yy_verify`
- `flex_int32_t yy_nxt`

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

- netsim/conf/lexer.c

3.51 yyalloc Union Reference

Collaboration diagram for yyalloc:



Data Fields

- yytype_int16 **yyss_alloc**
- **YYSTYPE yyvs_alloc**
- **YYLTYPE yyls_alloc**

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

- netsim/conf/parser.c

3.52 YYLTYPE Struct Reference

Data Fields

- int **first_line**
- int **first_column**
- int **last_line**
- int **last_column**

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

- netsim/conf/parser.c
- netsim/conf/parser.h

3.53 YYSTYPE Union Reference

Data Fields

- char * **str**
- int **ival**
- double **dval**

The documentation for this union was generated from the following files:

- netsim/conf/parser.c
- netsim/conf/parser.h

Chapter 4

File Documentation

4.1 def.h File Reference

Data Structures

- union `time`

Defines

- #define `MAX_INT` INT_MAX

Typedefs

- typedef union `time` TIME

4.1.1 Detailed Description

Date

Created on: Apr 8, 2011

Author

iizke

4.1.2 Typedef Documentation

4.1.2.1 `typedef union time TIME`

Time definition

4.2 error.c File Reference

```
#include <stdlib.h>
#include "error.h"
#include "list/linked_list.h"
```

Functions

- void [error](#) (char *msg, FILE *fp)
- int [trash_collect_garbage](#) ([GARBAGE](#) *g)
- void * [gc_malloc](#) (int size)
- int [trash_clean](#) ()
- int [trash_init](#) ()

Variables

- [TRASH trash](#)

4.2.1 Detailed Description

Date

Created on: Jun 5, 2011

Author

iizke

4.2.2 Function Documentation

4.2.2.1 void [error](#) (char * msg, FILE * fp)

Print out error message to file

Parameters

<i>msg</i>	: Message
<i>fp</i>	: file pointer

4.2.2.2 void* [gc_malloc](#) (int size)

Malloc with garbage collector

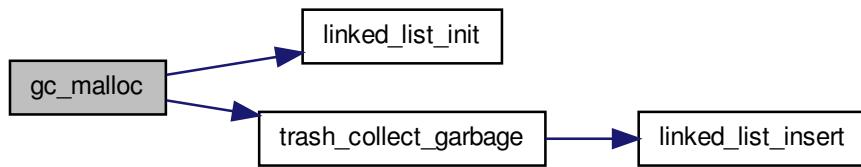
Parameters

<i>size</i>	: size of memory
-------------	------------------

Returns

pointer to new allocated memory

Here is the call graph for this function:

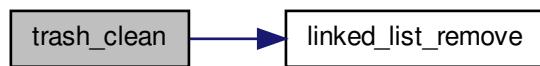
**4.2.2.3 int trash_clean()**

Empty trash

Returns

Error code

Here is the call graph for this function:

**4.2.2.4 int trash_collect_garbage(GARBAGE * g)**

Update new malloc link to garbage

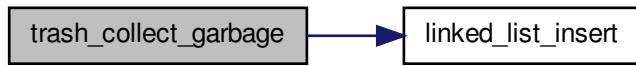
Parameters

<i>g</i>	: garbage link
----------	----------------

Returns

Error code

Here is the call graph for this function:



4.3 error.h File Reference

```
#include <stdio.h>
#include "list/linked_list.h"
```

Data Structures

- struct [trash](#)
- struct [garbage](#)

Defines

- #define [DEBUG](#)
Turn on Debug mode.
- #define [LEVEL_ERROR](#) 0b00000001
Print out all error-labeled messages.
- #define [LEVEL_WARNING](#) 0b00000010
Print out all warning-labeled messages.
- #define [LEVEL_INFO](#) 0b00000100
Print out all info-labeled messages.
- #define [LEVEL_EW](#) 0b00000011
Print out all messages labeled with Error and Warning.
- #define [LEVEL_EI](#) 0b00000101
Print out all messages labeled with Error and Info.

- #define **LEVEL_EWI** 0b00000111

LEVEL_EWI Print out all messages labeled with Error, Info, Warning.

- #define **iprint**(level, fmt, args...)
- #define **try**(stm)
- #define **check_null_pointer**(_p)
- #define **TRUE** 1
- #define **FALSE** 0
- #define **SUCCESS** 0

Return this value when a function finishes successfully.

- #define **ERR_POINTER_NULL** (-1)

Error when a pointer is null.

- #define **ERR_MALLOC_FAIL** (-2)

Error when not enough memory in system.

- #define **ERR_RANDOM_TYPE_FAIL** (-17)

Error when user does not provide a correct Flow Type ID.

- #define **ERR_EVENT_TIME_WRONG** (-18)

Error when the time of event is not consistent, eg. it is late than the current time.

- #define **ERR_EVENT_TYPE_FAIL** (-19)

Error when an event-type is not supported.

- #define **ERR_PACKET_STATE_WRONG** (-20)

Error when packet state is not correct.

- #define **ERR_CSMA_INCONSISTENT** (-30)

Error while inconsistent happen in simulating CSMA.

- #define **imalloc**(size) gc_malloc(size)
- #define **malloc_gc**(size) gc_malloc(size)

Typedefs

- typedef struct **trash** **TRASH**
- typedef struct **garbage** **GARBAGE**

Functions

- void * **gc_malloc** (int size)
- void **error** (char *msg, FILE *fp)
- int **trash_collect_garbage** (**GARBAGE** *g)
- int **trash_clean** ()
- int **trash_init** ()

Variables

- long **debug**

4.3.1 Detailed Description

Define some error codes and some utility functions/macros to announce errors.

Date

Created on: Apr 6, 2011

Author

iizke

4.3.2 Define Documentation

4.3.2.1 #define check_null_pointer(_p)

Value:

```
{
    if (!(_p)) {
        iprint(LEVEL_WARNING, "NULL pointer\n");
        return ERR_POINTER_NULL;
    }
}
```

4.3.2.2 #define iprint(level, fmt, args...)

Value:

```
{
    if (level & debug) {
        printf("File %s, line %d: \n", __FILE__, __LINE__);
        printf(fmt, ## args);
    }
}
```

4.3.2.3 #define try(*stm*)**Value:**

```
{
    int _err_ = stm;
    if (_err_ < 0)
        return _err_;
}
```

4.3.3 Function Documentation

4.3.3.1 void error (*char * msg, FILE * fp*)

Print out error message to file

Parameters

<i>msg</i>	: Message
<i>fp</i>	: file pointer

4.3.3.2 void* gc_malloc (*int size*)

Malloc with garbage collector

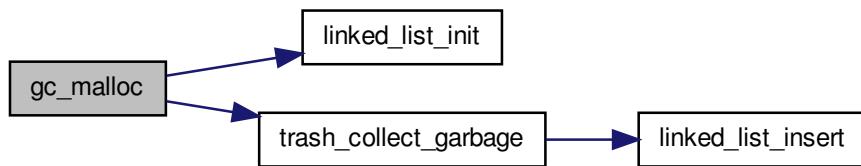
Parameters

<i>size</i>	: size of memory
-------------	------------------

Returns

pointer to new allocated memory

Here is the call graph for this function:



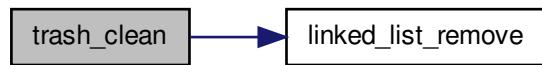
4.3.3.3 int trash_clean()

Empty trash

Returns

Error code

Here is the call graph for this function:

**4.3.3.4 int trash_collect_garbage (GARBAGE * g)**

Update new malloc link to garbage

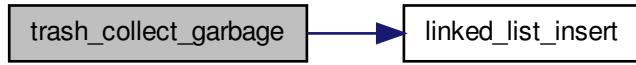
Parameters

<i>g</i>	: garbage link
----------	----------------

Returns

Error code

Here is the call graph for this function:



4.4 irand/irand.h File Reference

Data Structures

- struct [weibull_params](#)
Parameters of Weibull distribution.
- struct [uniform_params](#)
Parameters of Uniform distribution.
- struct [empirical_params](#)
Parameters of Empirical Discrete distribution.
- struct [random_distribution](#)
Random distribution structure (a framework)

Defines

- #define [RND_TYPE_LINEAR](#) 1
Random generator: Linear Congruential.
- #define [RND_TYPE_COMBINED](#) 2
Random generator: Combined Linear Congruential.
- #define [RND_TYPE_TAUSWORTHE](#) 3
Random generator: Tausworthe (Not supported right now)

Typedefs

- typedef struct [random_distribution](#) RANDOM_DIST
Random distribution structure (a framework)

Functions

- int [irand_init](#) ()
- int [irand_new_seed](#) (unsigned long seed)
- int [irand_random_seed](#) ()
- unsigned long [irand_gen_random](#) (int type)
- unsigned long [irand_gen_srandom](#) (int type, unsigned long seed)
- double [irand_gen_random_real](#) (int type)
- double [irand_gen_srandom_real](#) (int type, unsigned long seed)

- unsigned long **irand_gen_range_random** (int type, unsigned long from, unsigned long to)
- double **irand_gen_erlang** (int order, double lambda)
- double **irand_gen_exp** (double lambda)
- double **irand_gen_uniform** (double from, double to)
- double **irand_gen_int_uniform** (double from, double to)
- double **irand_gen_bernoulli** (double prob)
- double **irand_gen_pareto** (double lambda)

4.4.1 Detailed Description

My implementation of Random number and distribution (called irand)

Date

Created on: Apr 28, 2011

Author

iizke

4.4.2 Function Documentation

4.4.2.1 double **irand_gen_bernoulli** (double *prob*)

Bernoulli generator of random number

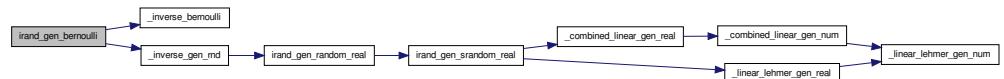
Parameters

<i>prob</i>	: probability of success event (value 1)
-------------	------------------------------------------

Returns

A random value (0 or 1)

Here is the call graph for this function:



4.4.2.2 double **irand_gen_erlang** (int *order*, double *lambda*)

Erlang Generator of random number

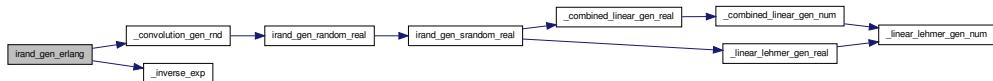
Parameters

<i>order</i>	: order of Erlang distribution
<i>lambda</i>	: the average rate of Erlang variable

Returns

A real random number of Erlang-k distribution

Here is the call graph for this function:

**4.4.2.3 double irand_gen_exp (double *lambda*)**

Exponential generator of random number

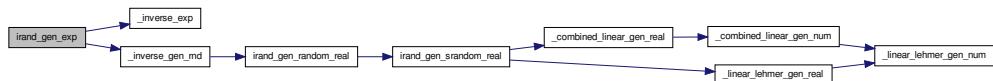
Parameters

<i>lambda</i>	: the average rate of random variable
---------------	---------------------------------------

Returns

A real random value

Here is the call graph for this function:

**4.4.2.4 double irand_gen_int_uniform (double *from*, double *to*)**

Integer Uniform generator of random number

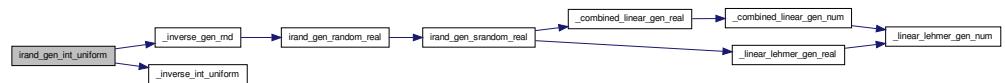
Parameters

<i>from</i>	: lower-bound value
<i>to</i>	: upper-bound value

Returns

A integer random value in range [from, to]

Here is the call graph for this function:

**4.4.2.5 double irand_gen_pareto (double *lambda*)**

Pareto generator of random number

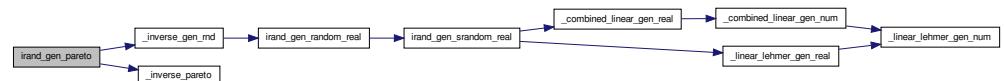
Parameters

<i>lambda</i>	: parameter of Pareto distribution
---------------	------------------------------------

Returns

A random number

Here is the call graph for this function:

**4.4.2.6 unsigned long irand_gen_random (int *type*)**

Pseudo Integer Random Uniform generator

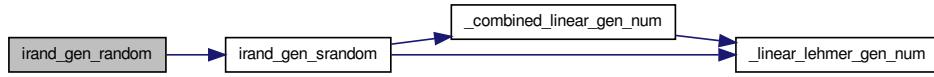
Parameters

<i>type</i>	: type of algorithm (linear congruential or combined linear congruential)
-------------	---------------------------------------------------------------------------

Returns

Random value

Here is the call graph for this function:



4.4.2.7 double irand_gen_random_real (int type)

Pseudo Real-value Random Uniform generator

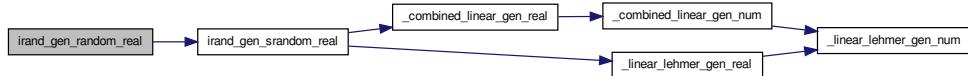
Parameters

<i>type</i>	: type of algorithm (linear congruential or combined linear congruential)
-------------	---------------------------------------------------------------------------

Returns

Random value

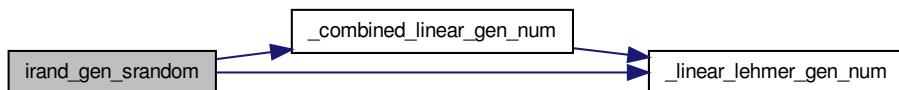
Here is the call graph for this function:



4.4.2.8 unsigned long irand_gen_srandom (int type, unsigned long seed)

Generate random number in Combined Linear Congruential

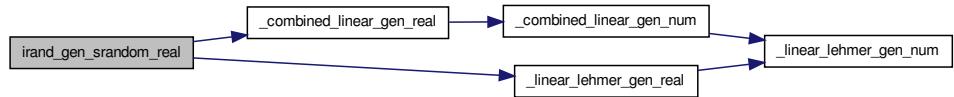
Here is the call graph for this function:



4.4.2.9 double irand_gen_random_real (int type, unsigned long seed)

Generate random number in Combined Linear Congruential

Here is the call graph for this function:



4.4.2.10 double irand_gen_uniform (double from, double to)

Uniform generator of random number (real value)

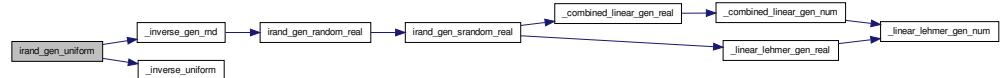
Parameters

<code>from</code>	: lower-bound value
<code>to</code>	: upper-bound value

Returns

A real random value in range [from, to]

Here is the call graph for this function:



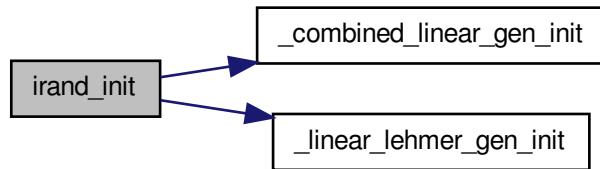
4.4.2.11 int irand_init ()

Initialize pseudo random generator

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.4.12 int irand_new_seed(unsigned long seed)

Pseudo random generator with seed

Parameters

<code>seed</code>	: seed of generator
-------------------	---------------------

Returns

integer pseudo random value

4.4.13 int irand_random_seed()

Pseudo random generator with random-selected seed

Returns

integer pseudo random value

Here is the call graph for this function:



4.5 irand/rnddist.c File Reference

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "irand.h"
#include "../error.h"
```

Functions

- double `_convolution_gen_rnd` (int n, double(*func)(double, void *), void *params)
- double `_inverse_gen_rnd` (double(*inverse_func)(double, void *), void *params)
- double `_inverse_empirical` (double p, void *params)
- double `_composition_gen_rnd` (int n, `RANDOM_DIST` *rnd, double *probs)
- double `_inverse_exp` (double p, void *params)
- double `_inverse_uniform` (double p, void *params)
- double `_inverse_int_uniform` (double p, void *params)
- double `_inverse_pareto` (double p, void *params)
- double `_inverse_weibull` (double p, void *params)
- double `_inverse_bernoulli` (double p, void *params)
- double `_inverse_geometric` (double p, void *params)
- double `irand_gen_erlang` (int order, double lambda)
- double `irand_gen_exp` (double lambda)
- double `irand_gen_uniform` (double from, double to)
- double `irand_gen_int_uniform` (double from, double to)
- double `irand_gen_bernoulli` (double prob)
- double `irand_gen_pareto` (double lambda)
- int `test_gen_distribution` ()

4.5.1 Detailed Description

Random number in a well-known distribution. Some methods are used:

- (1) Inverse Transform method
- (2) Convolution method
- (3) Acceptance/Rejection technique (not implemented)
- (4) Composition method
- (5) Other (Normal, Poisson, Binomial) - not implemented

Date

Created on: May 3, 2011

Author

iizke

4.5.2 Function Documentation

4.5.2.1 double _composition_gen_rnd (int *n*, RANDOM_DIST * *rnd*, double * *probs*)

Framework of generating random variable based on composition method.

$$\text{CDF}(X) = p_1 \cdot \text{CDF}_1(X) + p_2 \cdot \text{CDF}_2(X) + \dots + p_n \cdot \text{CDF}_n(X)$$

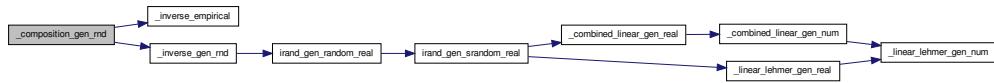
Parameters

<i>n</i>	: Number of random variables.
<i>rnd</i>	: List of random variables with specification of their distribution.
<i>probs</i>	: List of probabilities for each random values.

Returns

A random value.

Here is the call graph for this function:



4.5.2.2 double _convolution_gen_rnd (int *n*, double(*)(double, void *) *func*, void * *params*)

Framework of generating random variable based on convolution method.

$$Y = X_1 + X_2 + \dots + X_n$$

X_1, X_2, \dots, X_n are i.i.d random variable.

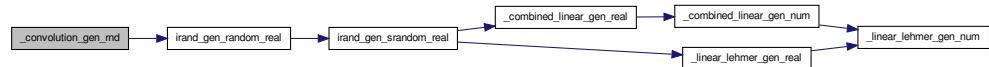
Parameters

<i>n</i>	: number of i.i.d random variables.
<i>func</i>	: Generated random function of each random variable.
<i>params</i>	: Parameters used for function func.

Returns

A random value.

Here is the call graph for this function:



4.5.2.3 double `_inverse_bernoulli` (double *p*, void * *params*)

Inverse Transform function of Bernoulli distribution

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: pointer to a value that is a probability of success (1)

Returns

A real random number of Bernoulli distribution

4.5.2.4 double `_inverse_empirical` (double *p*, void * *params*)

Inverse Transform function of Empirical Discrete distribution.

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: parameters of this distribution includes: probabilities and appropriate values. If list of value is not determined (NULL), then the return value will be the index.

Returns

A real random number of this distribution

4.5.2.5 double `_inverse_exp` (double *p*, void * *params*)

Inverse Transform function of Exponential distribution

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: structure that contains lambda (parameter of exponential distribution)

Returns

A real random number of exponential distribution

4.5.2.6 double _inverse_gen_rnd (double(*)(double, void *) *inverse_func*, void * *params*)

Framework of generating a random value (of distribution) based on Inverse Transform method

Parameters

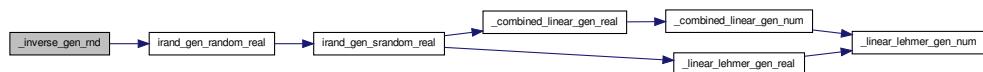
<i>inverse_func</i>	: inverse function (of a particular distribution).
<i>params</i>	: parameters used for function <i>inverse_func</i> .

Returns

random value.

1st step: Generate random integer number

Here is the call graph for this function:

**4.5.2.7 double _inverse_geometric (double *p*, void * *params*)**

Inverse Transform function of Geometric distribution

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: structure that contains parameter of Geometric distribution

Returns

A real random number of Geometric distribution

4.5.2.8 double _inverse_int_uniform (double *p*, void * *params*)

Inverse Transform function of Discrete (integer) Uniform distribution

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: structure that contains parameter of uniform distribution

Returns

A integer random number of uniform distribution

4.5.2.9 double _inverse_pareto (double *p*, void * *params*)

Inverse Transform function of Pareto distribution

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: structure that contains lambda (parameter of pareto distribution)

Returns

A real random number of Pareto distribution

4.5.2.10 double _inverse_uniform (double *p*, void * *params*)

Inverse Transform function of Uniform distribution

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: structure that contains parameter of uniform distribution

Returns

A real random number of uniform distribution

4.5.2.11 double _inverse_weibull (double *p*, void * *params*)

Inverse Transform function of Weibull distribution

Parameters

<i>p</i>	: a uniform random number
<i>params</i>	: structure that contains parameter of Weibull distribution

Returns

A real random number of Weibull distribution

4.5.2.12 double irand_gen_bernoulli (double *prob*)

Bernoulli generator of random number

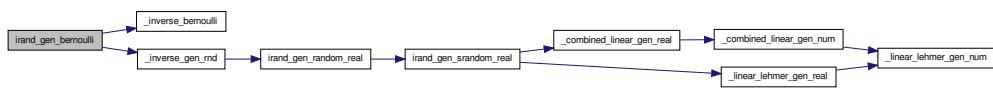
Parameters

<i>prob</i>	: probability of success event (value 1)
-------------	------------------------------------------

Returns

A random value (0 or 1)

Here is the call graph for this function:

**4.5.2.13 double irand_gen_erlang (int *order*, double *lambda*)**

Erlang Generator of random number

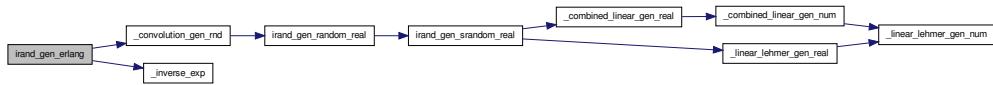
Parameters

<i>order</i>	: order of Erlang distribution
<i>lambda</i>	: the average rate of Erlang variable

Returns

A real random number of Erlang-k distribution

Here is the call graph for this function:

**4.5.2.14 double irand_gen_exp (double *lambda*)**

Exponential generator of random number

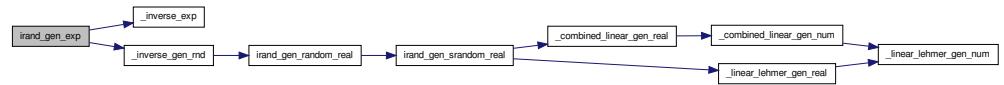
Parameters

<i>lambda</i>	: the average rate of random variable
---------------	---------------------------------------

Returns

A real random value

Here is the call graph for this function:

**4.5.2.15 double irand_gen_int_uniform (double *from*, double *to*)**

Integer Uniform generator of random number

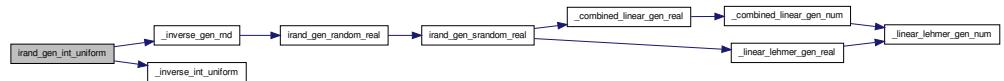
Parameters

<i>from</i>	: lower-bound value
<i>to</i>	: upper-bound value

Returns

A integer random value in range [from, to]

Here is the call graph for this function:

**4.5.2.16 double irand_gen_pareto (double *lambda*)**

Pareto generator of random number

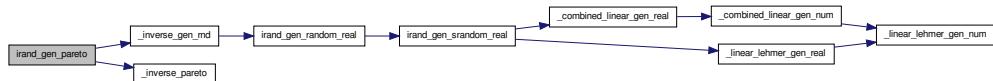
Parameters

<i>lambda</i>	: parameter of Pareto distribution
---------------	------------------------------------

Returns

A random number

Here is the call graph for this function:



4.5.2.17 double irand_gen_uniform (double from, double to)

Uniform generator of random number (real value)

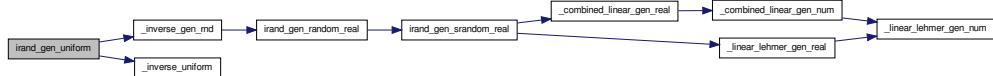
Parameters

<i>from</i>	: lower-bound value
<i>to</i>	: upper-bound value

Returns

A real random value in range [from, to]

Here is the call graph for this function:



4.6 irand/rndnum.c File Reference

```

#include <stdio.h>
#include <time.h>
#include "../error.h"
#include "irand.h"
  
```

Data Structures

- struct [linear_congruential_gen](#)
- struct [combined_linear_congruential_gen](#)

TypeDefs

- `typedef struct linear_congruential_gen LINEAR_LEHMER_GEN`
- `typedef struct combined_linear_congruential_gen COMBINED_LINEAR_GEN`

Functions

- `unsigned long _linear_lehmer_gen_num (LINEAR_LEHMER_GEN *gen)`
- `double _linear_lehmer_gen_real (LINEAR_LEHMER_GEN *gen)`
- `int _linear_lehmer_gen_init (LINEAR LEHMER_GEN *gen)`
- `unsigned long _combined_linear_gen_num (COMBINED_LINEAR_GEN *gen)`
- `double _combined_linear_gen_real (COMBINED_LINEAR_GEN *gen)`
- `int _combined_linear_gen_init (COMBINED_LINEAR_GEN *gen)`
- `unsigned long irand_gen_random (int type)`
- `double irand_gen_random_real (int type)`
- `unsigned long irand_gen_srandom (int type, unsigned long seed)`
- `double irand_gen_srandom_real (int type, unsigned long seed)`
- `unsigned long irand_gen_range_random (int type, unsigned long from, unsigned long to)`
- `int irand_init ()`
- `int irand_new_seed (unsigned long seed)`
- `int irand_random_seed ()`

Variables

- `LINEAR_LEHMER_GEN linear_rnd_gen`
Pseudo Random generator using Linear Congruential technique.
- `COMBINED_LINEAR_GEN combined_rnd_gen`
Pseudo Random generator using Combined Linear Congruential technique.

4.6.1 Detailed Description

Pseudo-Random number generators. Some methods are used:

- (1) Linear Congruential
- (2) Combined Linear Congruential
- (3) TAUSWORTHE (not implemented)

Date

Created on: Apr 20, 2011

Author

iizke

4.6.2 Typedef Documentation

4.6.2.1 `typedef struct combined_linear_congruential_gen COMBINED_LINEAR_GEN`

Random number generator based on Combined-Linear-Congruential Generator with two Linear Congruential Generator

4.6.2.2 `typedef struct linear_congruential_gen LINEAR_LEHMER_GEN`

Linear Congruential Generator

4.6.3 Function Documentation

4.6.3.1 `int _combined_linear_gen_init(COMBINED_LINEAR_GEN *gen)`

Initialize Combined Linear Congruential Generator

Parameters

<code>gen</code>	: Combined Linear Congruential Generator structure
------------------	----------------------------------------------------

Returns

Error code (defined libs/error.h)

4.6.3.2 `unsigned long _combined_linear_gen_num(COMBINED_LINEAR_GEN *gen)`

Generate a pseudo random number based on Combined Linear Congruential Generator

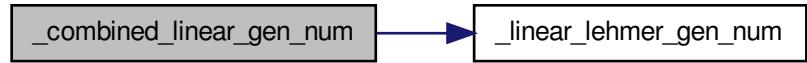
Parameters

<code>gen</code>	: Combined Linear Congruential Generator
------------------	------------------------------------------

Returns

Random number

Here is the call graph for this function:



4.6.3.3 double _combined_linear_gen_real(COMBINED_LINEAR_GEN * gen)

generate real number in range [0,1]

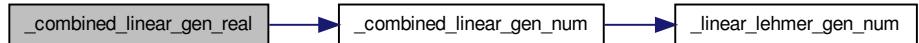
Parameters

<i>gen</i>	: Combined Linear Generator
------------	-----------------------------

Returns

real number

Here is the call graph for this function:



4.6.3.4 int _linear_lehmer_gen_init(LINEAR_LEHMER_GEN * gen)

Initialize the Linear Congruential Generator

Parameters

<i>gen</i>	: Linear Congruential Generator
------------	---------------------------------

Returns

Error code (defined in libs/error.h)

Module = $2^{31} - 1 = 2147483647$

Multiplier = $7^5 = 16807$

Increment = 0

4.6.3.5 `unsigned long _linear_lehmer_gen_num (LINEAR_LEHMER_GEN * gen)`

Linear Congruential Generator

Parameters

<code>gen</code>	(generator)
------------------	-------------

Returns

Random number

4.6.3.6 `double _linear_lehmer_gen_real (LINEAR_LEHMER_GEN * gen)`

Linear Congruential Generator

Parameters

<code>gen</code>	(generator)
------------------	-------------

Returns

Random number

Here is the call graph for this function:



4.6.3.7 `unsigned long irand_gen_random (int type)`

Pseudo Integer Random Uniform generator

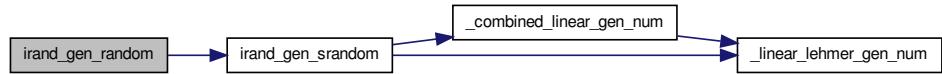
Parameters

<code>type</code>	: type of algorithm (linear congruential or combined linear congruential)
-------------------	---------------------------------------------------------------------------

Returns

Random value

Here is the call graph for this function:

**4.6.3.8 double irand_gen_random_real (int type)**

Pseudo Real-value Random Uniform generator

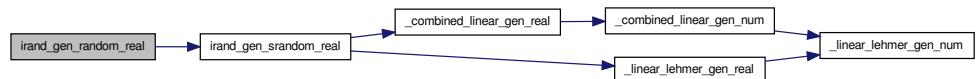
Parameters

<i>type</i>	: type of algorithm (linear congruential or combined linear congruential)
-------------	---------------------------------------------------------------------------

Returns

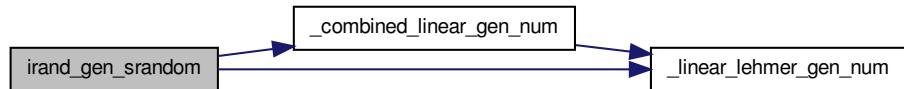
Random value

Here is the call graph for this function:

**4.6.3.9 unsigned long irand_gen_srandom (int type, unsigned long seed)**

Generate random number in Combined Linear Congruential

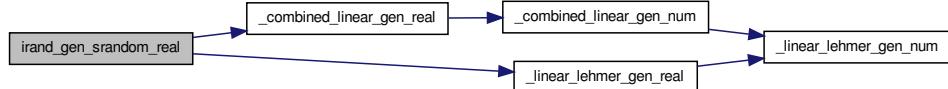
Here is the call graph for this function:



4.6.3.10 double irand_gen_srandom_real (int type, unsigned long seed)

Generate random number in Combined Linear Congruential

Here is the call graph for this function:



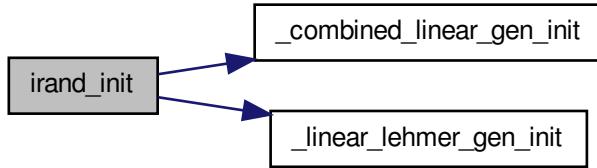
4.6.3.11 int irand_init ()

Initialize pseudo random generator

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.6.3.12 int irand_new_seed (unsigned long seed)

Pseudo random generator with seed

Parameters

<i>seed</i>	: seed of generator
-------------	---------------------

Returns

integer pseudo random value

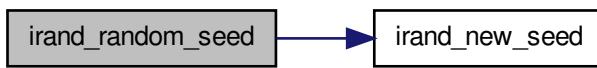
4.6.3.13 int irand_random_seed ()

Pseudo random generator with random-selected seed

Returns

integer pseudo random value

Here is the call graph for this function:



4.7 list/heap.c File Reference

4.7.1 Detailed Description

Heap implementation

Date

Created on: Jun 6, 2011

Author

iizke

4.8 list/heap.h File Reference

Data Structures

- struct [heap](#)

Typedefs

- typedef struct [heap](#) [HEAP](#)

Functions

- int [heap_init](#) ([HEAP](#) **)
- int [heap_setup](#) ([HEAP](#) *)
- int [heap_insert](#) ([HEAP](#) *, void *, int)
- int [heap_remove](#) ([HEAP](#) *, void *, int)

4.8.1 Detailed Description

Heap structure

Date

Created on: Jun 6, 2011

Author

iizke

4.9 list/linked_list.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "linked_list.h"
#include "../error.h"
```

Functions

- int `linked_list_init (LINKED_LIST *l)`
- int `linked_list_insert (LINKED_LIST *l, LINKED_LIST *e)`
- int `linked_list_remove (LINKED_LIST *e)`
- int `linked_list_get_first (LINKED_LIST *l, LINKED_LIST **e)`
- int `print_list (LINKED_LIST *l)`
- int `test_linked_list ()`
- int `linked_list_man_init (LINKED_LIST_MAN *lm)`
- int `linked_list_man_init_conf (LINKED_LIST_MAN *lm, int conf)`
- int `linked_list_man_insert (LINKED_LIST_MAN *lm, LINKED_LIST *e)`
- int `linked_list_man_remove (LINKED_LIST_MAN *lm, LINKED_LIST *e)`
- int `linked_list_man_get_first (LINKED_LIST_MAN *lm, LINKED_LIST **e)`
- int `linked_list_man_get_free_entry (LINKED_LIST_MAN *lm, LINKED_LIST **e)`
- int `linked_list_man_alloc (LINKED_LIST_MAN *lm, LINKED_LIST **e, int size)`
- int `test_linked_list_man ()`

4.9.1 Detailed Description

Double linked list functions

Date

Created on: Apr 6, 2011

Author

iizke

4.9.2 Function Documentation

4.9.2.1 int `linked_list_get_first (LINKED_LIST * l, LINKED_LIST ** e)`

Return the first element in a linked list

Parameters

<code>l</code>	: Linked list
<code>e</code>	: returned the first element in list

Generated on Mon Jun 6 2011 01:02:57 for QILFAU by Doxygen

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.9.2.2 int linked_list_init (LINKED_LIST * l)

Initialize linked list structure

Parameters

<i>l</i>	: Linked list
----------	---------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.9.2.3 int linked_list_insert (LINKED_LIST * l, LINKED_LIST * e)

Insert new element into a linked list. Element can be seen as a linked list

Parameters

<i>l</i>	: Linked list
<i>e</i>	: New element

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.9.2.4 int linked_list_man_alloc (LINKED_LIST_MANAGER * lm, LINKED_LIST ** e, int size)

Allocate a new linked list structure but controlled by a manager. The manager can reused the free memory (last allocation) for this allocation (no need to do malloc).

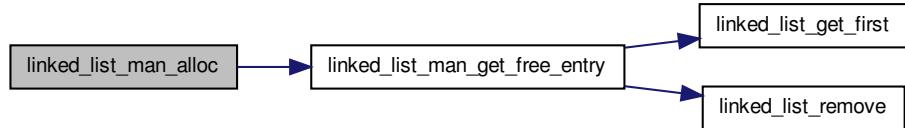
Parameters

<i>lm</i>	: Linked list manager
<i>e</i>	: new element
<i>size</i>	: size of new element. This parameter is only used when do malloc. So please take care of using this function: all new structures of linked list have the same size in your program.

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.9.2.5 int linked_list_man_get_first (LINKED_LIST_MANAGER * lm, LINKED_LIST ** e)

Get the first element in linked list controlled by a manager

Parameters

<i>lm</i>	: Linked list manager
<i>e</i>	: linked list element

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.9.2.6 int linked_list_man_get_free_entry (LINKED_LIST_MANAGER * lm, LINKED_LIST ** e)

Get an element which is freed before

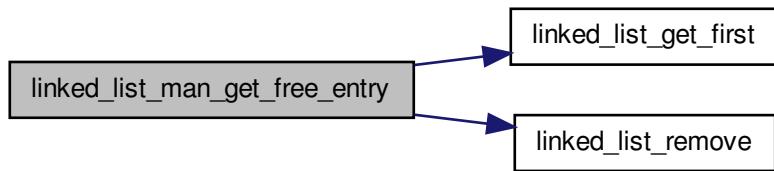
Parameters

<i>lm</i>	: Linked list manager
<i>e</i>	: free element

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.9.2.7 int linked_list_man_init(LINKED_LIST_MAN * lm)**

Initialize a manager of linked-list

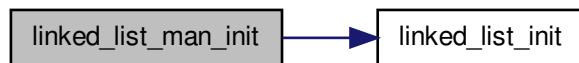
Parameters

<i>lm</i>	: linked list manager
-----------	-----------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.9.2.8 int linked_list_man_init_conf(LINKED_LIST_MAN * lm, int conf)**

Initialize linked list manager with configuration

Parameters

<i>lm</i>	: Linked list manager
<i>conf</i>	: Configuration. It can be LL_CONF_STORE_FREE or LL_CONF_STORE_ENTRY (or both)

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.9.2.9 int linked_list_man_insert(LINKED_LIST_MAN * lm, LINKED_LIST * e)**

Insert new element in a Linked list controlled by a manager

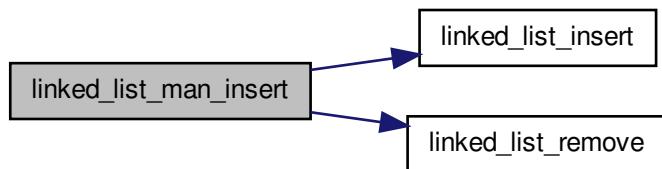
Parameters

<i>lm</i>	: Linked list manager
<i>e</i>	: new element

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.9.2.10 int linked_list_man_remove (LINKED_LIST_MAN * lm, LINKED_LIST * e)

Remove an element out of linked list controlled by a manager

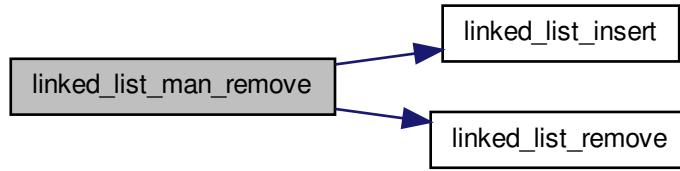
Parameters

<i>lm</i>	: Linked List manager
<i>e</i>	: Removed element. This element is not freed but reused in the next allocation

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.9.2.11 int linked_list_remove (LINKED_LIST * e)

Remove an element out of linked list

Parameters

<i>e</i>	: Removed element
----------	-------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.9.2.12 int print_list (LINKED_LIST * l)

Print out all elements of a linked list. Normally used in testing

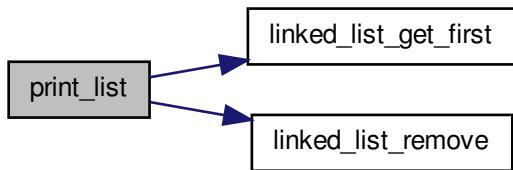
Parameters

<i>l</i>	: Linked list
----------	---------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.10 list/linked_list.h File Reference

```
#include <stddef.h>
```

Data Structures

- struct [linked_list](#)
- struct [linked_list_manager](#)

Defines

- #define [LL_CONF_STORE_FREE](#) 0b00000001
Configure list to allow storing free nodes.
- #define [LL_CONF_STORE_ENTRY](#) 0b00000010
Configure list to allow storing nodes in list.
- #define [ERR_LL_MAN_ALLOC](#) (-2)
Error code when failed in allocating new memory for node.
- #define [ERR_LL_CONF_WRONG](#) (-3)
Error code when configuration of linked list is not correct.
- #define [linked_list_is_empty](#)(_l) ((_l)->next == (_l))
Check linked list empty (return 1) or not (return 0)

- `#define container_of(ptr, type, member)`

Return the pointer of struct TYPE having a member name MEMBER with pointer ptr.

Typedefs

- `typedef struct linked_list LINKED_LIST`
- `typedef struct linked_list_manager LINKED_LIST_MAN`

Functions

- `int linked_list_init (LINKED_LIST *l)`
- `int linked_list_insert (LINKED_LIST *l, LINKED_LIST *e)`
- `int linked_list_remove (LINKED_LIST *e)`
- `int linked_list_get_first (LINKED_LIST *l, LINKED_LIST **e)`
- `int print_list (LINKED_LIST *l)`
- `int test_linked_list ()`
- `int linked_list_man_init (LINKED_LIST_MAN *lm)`
- `int linked_list_man_init_conf (LINKED_LIST_MAN *lm, int conf)`
- `int linked_list_man_insert (LINKED_LIST_MAN *lm, LINKED_LIST *e)`
- `int linked_list_man_remove (LINKED_LIST_MAN *lm, LINKED_LIST *e)`
- `int linked_list_man_get_first (LINKED_LIST_MAN *l, LINKED_LIST **e)`
- `int linked_list_man_get_free_entry (LINKED_LIST_MAN *lm, LINKED_LIST **e)`
- `int linked_list_man_alloc (LINKED_LIST_MAN *l, LINKED_LIST **e, int size)`
- `int test_linked_list_man ()`

4.10.1 Detailed Description

Double Linked List structure

Date

Created on: Apr 6, 2011

Author

iizke

4.10.2 Define Documentation

4.10.2.1 `#define container_of(ptr, type, member)`

Value:

```
({
    \n
    const typeof( ((type *)0)->member ) *__mptr = (ptr);      \
    (type *) ( (char *)__mptr - offsetof(type,member) );})
```

Return the pointer of struct TYPE having a member name MEMBER with pointer ptr.

4.10.3 Typedef Documentation

4.10.3.1 **typedef struct linked_list LINKED_LIST**

Linked list structure

4.10.3.2 **typedef struct linked_list_manager LINKED_LIST_MAN**

Linked list manager. Manage not only nodes of list but also free nodes (used for allocating new node)

4.10.4 Function Documentation

4.10.4.1 **int linked_list_get_first(LINKED_LIST *l, LINKED_LIST **e)**

Return the first element in a linked list

Parameters

<i>l</i>	: Linked list
<i>e</i>	: returned the first element in list

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.10.4.2 **int linked_list_init(LINKED_LIST *l)**

Initialize linked list structure

Parameters

<i>l</i>	: Linked list
----------	---------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.10.4.3 **int linked_list_insert(LINKED_LIST *l, LINKED_LIST *e)**

Insert new element into a linked list. Element can be seen as a linked list

Parameters

<i>l</i>	: Linked list
<i>e</i>	: New element

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.10.4.4 int linked_list_man_alloc (LINKED_LIST_MAN * *lm*, LINKED_LIST ** *e*, int *size*)

Allocate a new linked list structure but controlled by a manager. The manager can reused the free memory (last allocation) for this allocation (no need to do malloc).

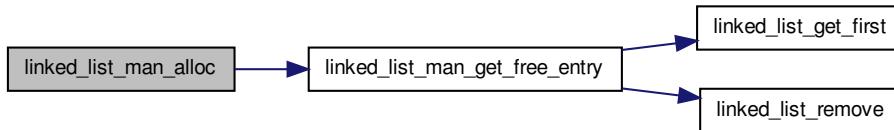
Parameters

<i>lm</i>	: Linked list manager
<i>e</i>	: new element
<i>size</i>	: size of new element. This parameter is only used when do malloc. So please take care of using this function: all new structures of linked list have the same size in your program.

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.10.4.5 int linked_list_man_get_first (LINKED_LIST_MAN * *lm*, LINKED_LIST ** *e*)

Get the first element in linked list controlled by a manager

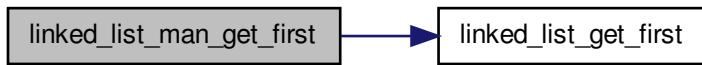
Parameters

<i>lm</i>	: Linked list manager
<i>e</i>	: linked list element

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.10.4.6 `int linked_list_man_get_free_entry (LINKED_LIST_MAN * lm,
LINKED_LIST ** e)`

Get an element which is freed before

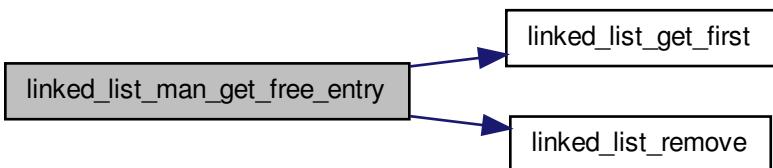
Parameters

<code>lm</code>	: Linked list manager
<code>e</code>	: free element

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.10.4.7 int linked_list_man_init(LINKED_LIST_MAN * *lm*)

Initialize a manager of linked-list

Parameters

<i>lm</i>	: linked list manager
-----------	-----------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.10.4.8 int linked_list_man_init_conf(LINKED_LIST_MAN * *lm*, int *conf*)**

Initialize linked list manager with configuration

Parameters

<i>lm</i>	: Linked list manager
<i>conf</i>	: Configuration. It can be LL_CONF_STORE_FREE or LL_CONF_STORE_ENTRY (or both)

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.10.4.9 int linked_list_man_insert(LINKED_LIST_MAN * lm, LINKED_LIST * e)

Insert new element in a Linked list controlled by a manager

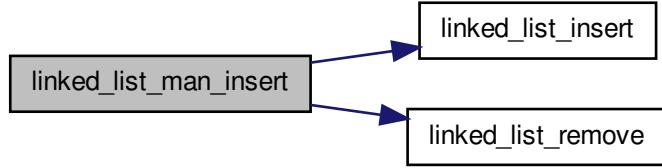
Parameters

<i>lm</i>	: Linked list manager
<i>e</i>	: new element

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.10.4.10 int linked_list_man_remove(LINKED_LIST_MAN * lm, LINKED_LIST * e)

Remove an element out of linked list controlled by a manager

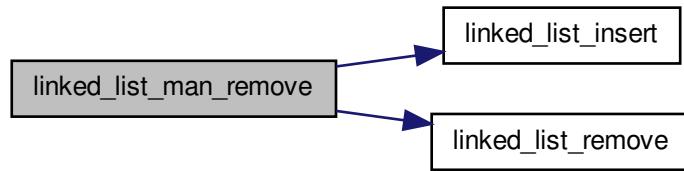
Parameters

<i>lm</i>	: Linked List manager
<i>e</i>	: Removed element. This element is not freed but reused in the next allocation

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.10.4.11 int linked_list_remove (LINKED_LIST * e)

Remove an element out of linked list

Parameters

e	: Removed element
---	-------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.10.4.12 int print_list (LINKED_LIST * l)

Print out all elements of a linked list. Normally used in testing

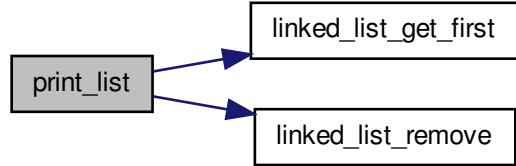
Parameters

l	: Linked list
---	---------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.11 netlib.c File Reference

```
#include <stdio.h>
#include "error.h"
#include "netsim/netsim.h"
#include "graph/main_graph.h"
#include "graph/main_rwa.h"
#include "matrix/matrix.h"
```

Functions

- int **check_netsim ()**
- int **check_matrix ()**
- int **main** (int nargs, char **args)

4.11.1 Detailed Description

Date

Created on: May 24, 2011

Author

iizke

4.12 netsim/conf/config.c File Reference

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

```
#include "../../error.h"
#include "../../random.h"
#include "parser.h"
#include "config.h"
```

Functions

- int `config_random_conf (RANDOM_CONF *rc)`
- int `config_init (CONFIG *conf)`
- int `config_setup (CONFIG *conf)`
- int `config_parse_file (char *f)`

Variables

- `CONFIG conf`

4.12.1 Detailed Description

Some functions related to config structure

Date

Created on: May 24, 2011

Author

iizke

4.12.2 Function Documentation

4.12.2.1 int config_init (CONFIG * *conf*)

Initialize the structure of CONFIG

Parameters

<code>conf</code>	: configuration
-------------------	-----------------

Returns

Error code (see more in `def.h` and `error.h`)

4.12.2.2 int config_parse_file (char * *f*)

Parse the configuration file (for example: test.conf)

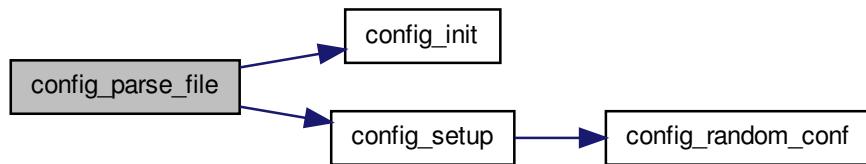
Parameters

<i>f</i>	: file name
----------	-------------

Returns

: Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.12.2.3 int config_random_conf (RANDOM_CONF * *rc*)**

Configure random distribution from parameters in RANDOM_CONFIG

Parameters

<i>rc</i>	: Random configuration
-----------	------------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.12.2.4 int config_setup (CONFIG * *conf*)

Configure random distribution in user configuration

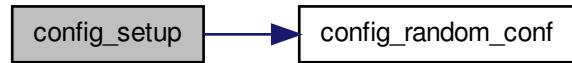
Parameters

<i>conf</i>	: User configuration
-------------	----------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.13 netsim/conf/config.h File Reference

```
#include <stdio.h>
#include "../random.h"
```

Data Structures

- struct random_config
- struct queue_config
- struct stop_config
- struct csma_conf
- struct config

Defines

- #define RANDOM_OTHER 1
- #define RANDOM_MARKOVIAN 2
- #define RANDOM_UNIFORM 3
- #define RANDOM_FILE 4
- #define RANDOM_BERNOULLI 5
- #define PROTOCOL_CSMA 0
- #define PROTOCOL_ONE_QUEUE 1
- #define STOP_QUEUE_ZERO 0
- #define STOP_QUEUE_NONZERO 1

Typedefs

- typedef struct random_config RANDOM_CONF
- typedef struct queue_config QUEUE_CONF
- typedef struct stop_config STOP_CONF
- typedef struct csma_conf CSMA_CONF
- typedef struct config CONFIG

Functions

- int `config_random_conf (RANDOM_CONF *rc)`
- int `config_init (CONFIG *conf)`
- int `config_setup (CONFIG *conf)`
- int `config_parse_file (char *file)`

4.13.1 Detailed Description

Define user configurations

Date

Created on: Apr 16, 2011

Author

iizke

4.13.2 Typedef Documentation

4.13.2.1 `typedef struct config CONFIG`

Configuration from user

4.13.2.2 `typedef struct queue_config QUEUE_CONF`

Queue configuration

4.13.2.3 `typedef struct random_config RANDOM_CONF`

Flow configuration is used to characterize a flow: what is its distribution, define some parameters.

4.13.2.4 `typedef struct stop_config STOP_CONF`

Define conditions of stopping program

4.13.3 Function Documentation

4.13.3.1 `int config_init (CONFIG * conf)`

Initialize the structure of CONFIG

Parameters

<code>conf</code>	: configuration
-------------------	-----------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.13.3.2 int config_parse_file (char * *f*)

Parse the configuration file (for example: test.conf)

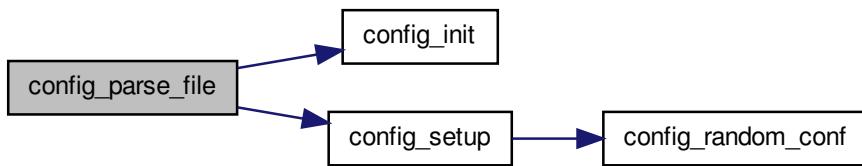
Parameters

<i>f</i>	: file name
----------	-------------

Returns

: Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.13.3.3 int config_random_conf (RANDOM_CONF * *rc*)**

Configure random distribution from parameters in RANDOM_CONFIG

Parameters

<i>rc</i>	: Random configuration
-----------	------------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.13.3.4 int config_setup (CONFIG * *conf*)

Configure random distribution in user configuration

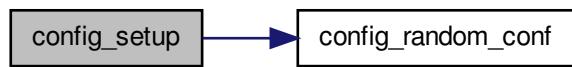
Parameters

<i>conf</i>	: User configuration
-------------	----------------------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.14 netsim/csma.c File Reference

```
#include <stdlib.h>
#include "conf/config.h"
#include "error.h"
#include "queues/fifo.h"
#include "random.h"
#include "csma.h"
```

Defines

- #define **queue_state**(qm) ((([FIFO_QINFO](#)*)(qm->curr_queue->info))->state)

Functions

- int [csma_remove_event](#) ([SYS_STATE_OPS](#) *ops, [EVENT](#) *e)
- [EVENT](#) * [csma_generate_arrival](#) ([CONFIG](#) *conf, [CSMA_STATE](#) *state)
- [EVENT](#) * [csma_generate_end_service](#) ([PACKET](#) *p, [CONFIG](#) *conf, [CSMA_STATE](#) *state)
- [EVENT](#) * [csma_generate_access](#) ([PACKET](#) *p, [CONFIG](#) *conf, [CSMA_STATE](#) *state)
- [EVENT](#) * [csma_generate_collision](#) ([PACKET](#) *p, [CONFIG](#) *conf, [CSMA_STATE](#) *state)

- int `csma_process_access_event` (EVENT *e, CONFIG *conf, CSMA_STATE *state)
- int `csma_process_arrival` (EVENT *e, CONFIG *conf, CSMA_STATE *state)
- int `csma_process_collision` (EVENT *e, CONFIG *conf, CSMA_STATE *state)
- int `csma_process_end_service` (EVENT *e, CONFIG *conf, CSMA_STATE *state)
- EVENT * `csma_generate_event` (int type, PACKET *p, CONFIG *conf, SYS_STATE_OPS *ops)
- int `csma_process_event` (EVENT *e, CONFIG *conf, SYS_STATE_OPS *ops)
- int `csma_get_next_event` (SYS_STATE_OPS *ops, EVENT **e)
- int `csma_allow_continue` (CONFIG *conf, SYS_STATE_OPS *ops)
- int `csma_state_init` (CSMA_STATE *state, CONFIG *conf)

4.14.1 Detailed Description

Model of CSMA system (Carrier Sense Multiple Access)

Date

Created on: May 16, 2011

Author

iizke

4.14.2 Function Documentation

4.14.2.1 int `int csma_allow_continue (CONFIG * conf, SYS_STATE_OPS * ops)`

Check whether system should be stopped or not.

Parameters

<code>conf</code>	: User configuration
<code>ops</code>	: Abstract system operators

Returns

0 if system should be stopped, 1 otherwise.

Here is the call graph for this function:



4.14.2.2 EVENT* csma_generate_access (PACKET * *p*, CONFIG * *conf*, CSMA_STATE * *state*)

Generate an access event (if necessary) or change to persistent-mode

Parameters

<i>p</i>	: packet
<i>conf</i>	: user configuration
<i>state</i>	: system state

Returns

New event (already added in the event list)

4.14.2.3 EVENT* csma_generate_arrival (CONFIG * *conf*, CSMA_STATE * *state*)

Generate arrival event

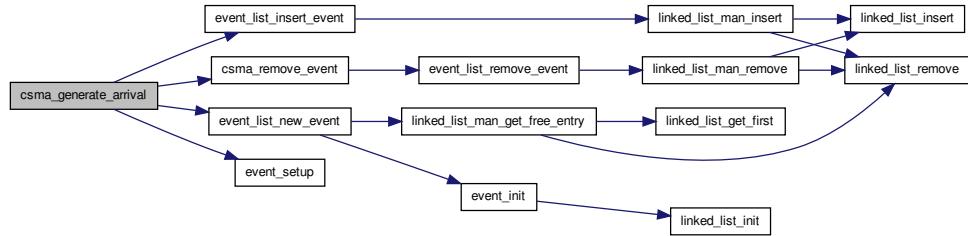
Parameters

<i>conf</i>	: user configuration
<i>state</i>	: system state

Returns

new event (already added in Event list)

Here is the call graph for this function:



4.14.2.4 EVENT* csma_generate_collision (PACKET * *p*, CONFIG * *conf*, CSMA_STATE * *state*)

Generate collision event

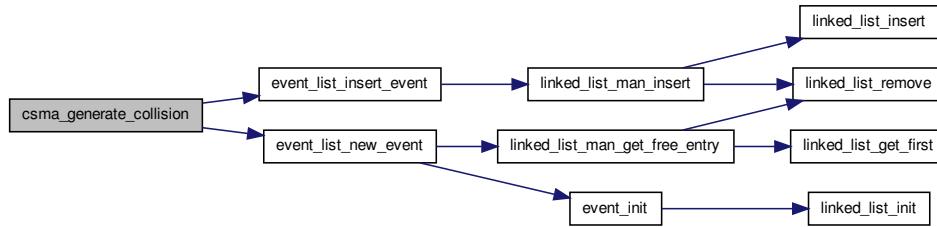
Parameters

<i>p</i>	: packet but should be NULL (no use)
<i>conf</i>	: user configuration
<i>state</i>	: system state

Returns

New event (already added in the event list)

Here is the call graph for this function:



4.14.2.5 EVENT* csma_generate_end_service (PACKET * *p*, CONFIG * *conf*, CSMA_STATE * *state*)

Generate new end-service event.

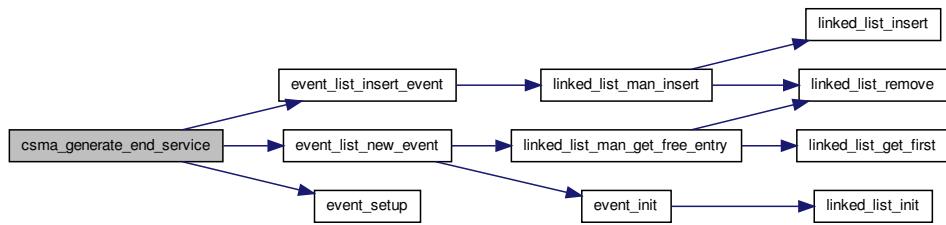
Parameters

<i>p</i>	: Processing packet (for this event)
<i>conf</i>	: user configuration
<i>state</i>	: System state

Returns

New event (already added in the event list)

Here is the call graph for this function:



4.14.2.6 EVENT* `csmagenerate_event` (int *type*, PACKET * *p*, CONFIG * *conf*, SYS_STATE_OPS * *ops*)

Generate new event.

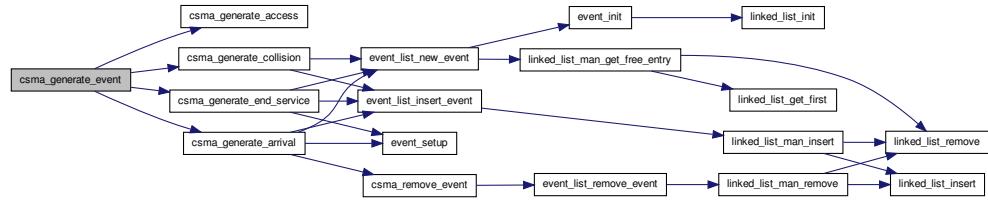
Parameters

<i>type</i>	: type of event
<i>p</i>	: Packet related to this new event
<i>conf</i>	: user configuration
<i>ops</i>	: Abstract system operations

Returns

New event

Here is the call graph for this function:



4.14.2.7 int csma_get_next_event (SYS_STATE_OPS * ops, EVENT ** e)

Get next event from event list

Parameters

<i>ops</i>	: Abstract system operations
<i>e</i>	: returned event

Returns

Error code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.14.2.8 int csma_process_access_event (EVENT * e, CONFIG * conf, CSMA_STATE * state)

Process Access event

Parameters

<i>e</i>	: Event
<i>conf</i>	: configuration from user
<i>state</i>	: system state

Returns

Error code (see more in [def.h](#) and [libs/error.h](#))

If channel is idle/free then occupy the channel

Here is the call graph for this function:



4.14.2.9 int csma_process_arrival (EVENT * e, CONFIG * conf, CSMA_STATE * state)

Process Arrival event

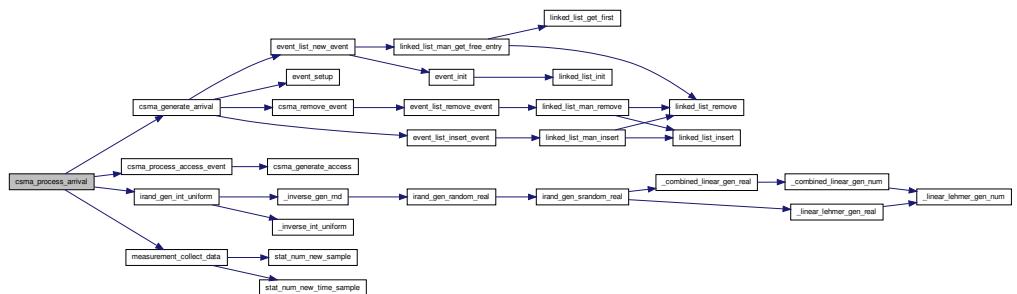
Parameters

<i>e</i>	: Event
<i>conf</i>	: configuration from user
<i>state</i>	: system state

Returns

Error code (see more in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.14.2.10 int csma_process_collision (EVENT * e, CONFIG * conf, CSMA_STATE * state)

Process Collision event

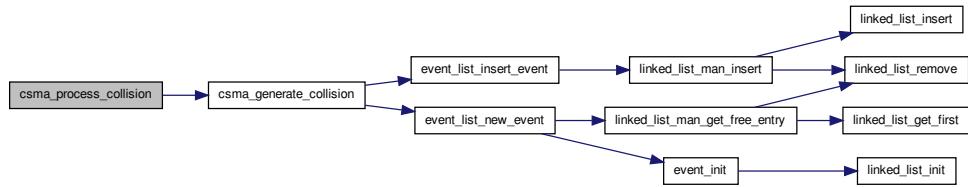
Parameters

<i>e</i>	: Event
<i>conf</i>	: configuration from user
<i>state</i>	: system state

Returns

Error code (see more in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.14.2.11 int csma_process_end_service (EVENT * e, CONFIG * conf, CSMA_STATE * state)

Process End-service event

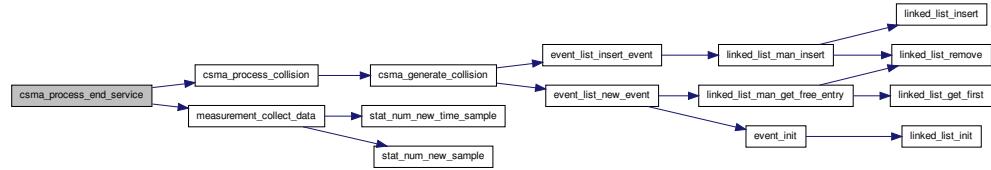
Parameters

<i>e</i>	: Event
<i>conf</i>	: configuration from user
<i>state</i>	: system state

Returns

Error code (see more in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.14.2.12 int csma_process_event (EVENT * e, CONFIG * conf, SYS_STATE_OPS * ops)

Process an event.

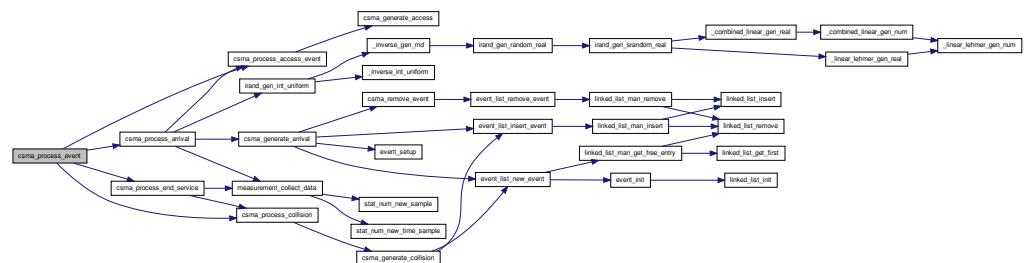
Parameters

<i>e</i>	: Event
<i>conf</i>	: User configuration
<i>ops</i>	:Abstract system operations

Returns

Error code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.14.2.13 int csma_remove_event (SYS_STATE_OPS * ops, EVENT * e)

Remove an event out of event list

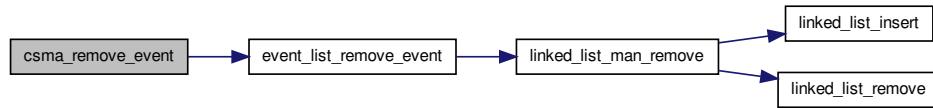
Parameters

<i>ops</i>	: Abstract system operations
<i>e</i>	: Event

Returns

Error code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.14.2.14 int csma_state_init(CSMA_STATE * state, CONFIG * conf)

Initialize CSMA system state

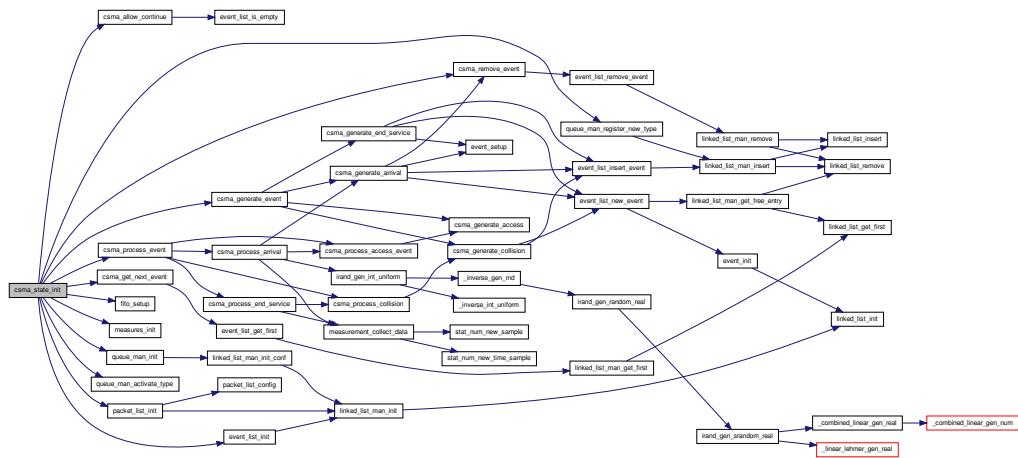
Parameters

<i>state</i>	: system state
<i>conf</i>	: user configuration

Returns

Error code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.15 netsim/csma.h File Reference

```
#include "event.h"
#include "queues/measures.h"
#include "netsim.h"
```

Data Structures

- struct `csma_state`

Defines

- #define `get_csma_state_from_ops(_ops)` (container_of(_ops, `CSMA_STATE`, ops))
- #define `CHANNEL_BUSY` 1
Channel state is Busy.
- #define `CHANNEL_FREE` 0
Channel state is Free.
- #define `QUEUE_STATE_NOPERSISTENT` 0
Queue state is Non-persistent CSMA.
- #define `QUEUE_STATE_PERSISTENT` 1
Queue state is persistent CSMA.
- #define `QUEUE_STATE_TRANSFERRING` 2
Queue is transferring packet.

Typedefs

- typedef struct `csma_state` `CSMA_STATE`

Functions

- int `csma_state_init (CSMA_STATE *state, CONFIG *conf)`

4.15.1 Detailed Description

CSMA system structure

Date

Created on: May 20, 2011

Author

iizke

4.15.2 Function Documentation**4.15.2.1 int csma_state_init (CSMA_STATE * state, CONFIG * conf)**

Initialize CSMA system state

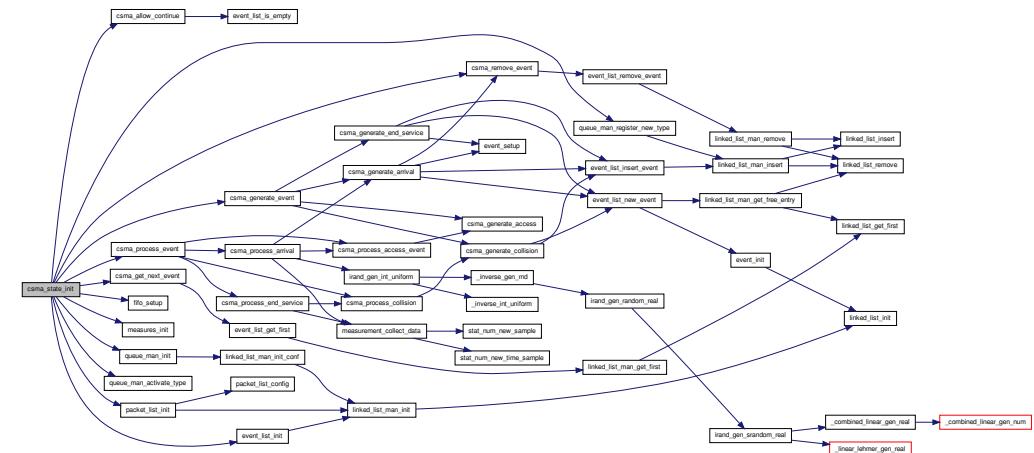
Parameters

<i>state</i>	: system state
<i>conf</i>	: user configuration

Returns

Error code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.16 netsim/event.c File Reference**

```
#include <stdlib.h>
#include "../error.h"
#include "event.h"
```

Functions

- int `event_list_init (EVENT_LIST *el)`
- int `event_list_new_event (EVENT_LIST *el, EVENT **e)`
- int `event_list_insert_event (EVENT_LIST *el, EVENT *e)`
- int `event_list_remove_event (EVENT_LIST *el, EVENT *e)`
- int `event_list_get_first (EVENT_LIST *el, EVENT **e)`
- int `event_list_is_empty (EVENT_LIST *l)`
- int `event_init (EVENT *e)`
- int `print_event_list (EVENT_LIST *l)`
- int `event_save (EVENT *e, FILE *f)`
- int `test_event_list_insert ()`

4.16.1 Detailed Description

Event and Event-list structure and related functions

Date

Created on: Apr 6, 2011

Author

iizke

4.16.2 Function Documentation

4.16.2.1 int event_init (EVENT * e)

Initialize parameters in Event structure

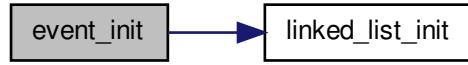
Parameters

<code>e</code>	: Event
----------------	---------

Returns

Error-code (defined in `def.h` and `libs/error.h`)

Here is the call graph for this function:



4.16.2.2 int event_list_get_first(EVENT_LIST * el, EVENT ** e)

Get the first event in the event-list

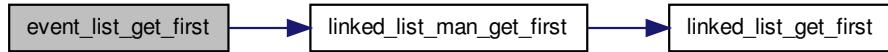
Parameters

<i>el</i>	: Event-list
<i>e</i>	: Output -> Event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.16.2.3 int event_list_init(EVENT_LIST * el)

Initialize parametters in EVENT_LIST

Parameters

<i>el</i>	: pointer to EVENT_LIST
-----------	-------------------------

Returns

Error-code (normal SUCCESS)

Here is the call graph for this function:



4.16.2.4 int event_list_insert_event(EVENT_LIST * el, EVENT * e)

Insert an event into event-list

Parameters

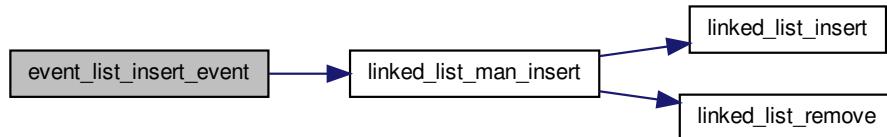
<i>el</i>	: Event list
<i>e</i>	: event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

sort from end of list, assume that [event_list](#) already sorted before

Here is the call graph for this function:



4.16.2.5 int event_list_is_empty(EVENT_LIST * l)

Check an Event-list empty or not

Parameters

<i>l</i>	: event list
----------	--------------

Returns

Return negative number if error. Return 1 if event list is empty, and return 0 otherwise.

4.16.2.6 int event_list_new_event (EVENT_LIST * el, EVENT ** e)

New event structure is allocated and init But this new event is not inserted into referenced Event-list

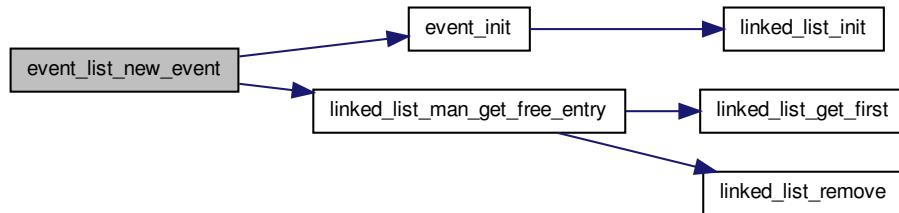
Parameters

<i>el</i>	: Event-list used for allocating new memory to Event structure
<i>e</i>	: output -> New event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:

**4.16.2.7 int event_list_remove_event (EVENT_LIST * el, EVENT * e)**

Remove an event out of event list. Note that, based on Linked-list-manager (LINKED_LIST_MAN), this event may not be freed but holding for future use (new-event).

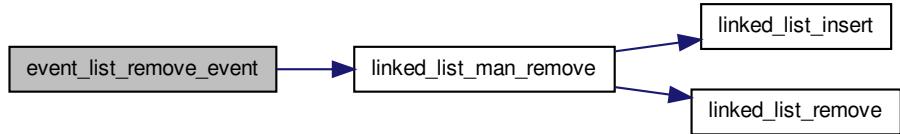
Parameters

<i>el</i>	: Event list
<i>e</i>	: removed event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.16.2.8 int event_save (EVENT * e, FILE * f)

Save an event information into file

Parameters

<i>e</i>	: Event
<i>f</i>	: File pointer

Returns

Error code (see more in file [def.h](#) and [libs/error.h](#))

4.16.2.9 int print_event_list (EVENT_LIST * l)

Print out to screen some info of every event in a list This info includes event-type, event-time

Parameters

<i>l</i>	: Event list
----------	--------------

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

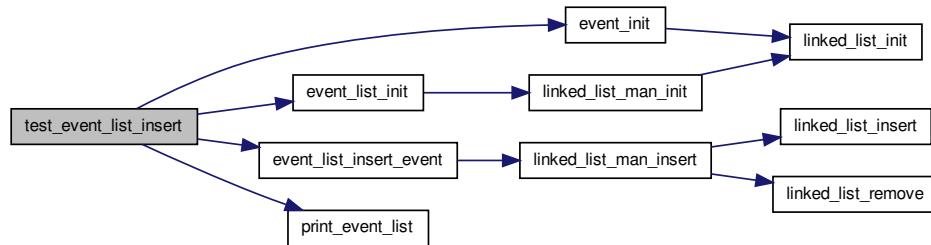
4.16.2.10 int test_event_list_insert ()

Unit test of function event_list_insert

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.17 netsim/event.h File Reference

```
#include <stdio.h>
#include "queues/queue.h"
```

Data Structures

- struct `event_info`
- struct `event`
- struct `event_list`

Defines

- #define `EVENT_ARRIVAL` 1
Arrival event.
- #define `EVENT_END_SERVICE` 2
End-of-Service event.
- #define `EVENT_ACCESS_CHANNEL` 3
Access Channel event.
- #define `EVENT_END_COLLISION` 4
End Collision event.
- #define `EVENT_SIGNAL_SOT` 5
Signal SoT (Start of Transmission) event.
- #define `EVENT_SIGNAL_EOT` 6

Signal EoT (End of Transmission) event.

- #define swap_prev_event(e2)
swap two adjacent event: this event and prev-event of this event

TypeDefs

- typedef struct event_info EVENTINFO
- typedef struct event EVENT
- typedef struct event_list EVENT_LIST

Functions

- int event_init (EVENT *e)
- int event_save (EVENT *e, FILE *file)
- int event_list_init (EVENT_LIST *el)
- int event_list_new_event (EVENT_LIST *el, EVENT **e)
- int event_list_insert_event (EVENT_LIST *el, EVENT *e)
- int event_list_remove_event (EVENT_LIST *el, EVENT *e)
- int event_list_get_first (EVENT_LIST *el, EVENT **e)
- int event_list_is_empty (EVENT_LIST *l)
- int test_event_list_insert ()

4.17.1 Detailed Description

Event structure

Date

Created on: Apr 6, 2011

Author

iizke

4.17.2 Define Documentation

4.17.2.1 #define swap_prev_event(e2)

Value:

```
{
    LINKED_LIST * _l1 = e2->list_node.prev;
    LINKED_LIST * _l2 = &e2->list_node;
    _l1->next = _l2->next;
    _l2->prev = _l1->prev;
    _l1->prev = _l2;
```

```

    _l2->next = _l1;
    _l1->next->prev = _l1;
    _l2->prev->next = _l2;
}

```

swap two adjacent event: this event and prev-event of this event

4.17.3 Typedef Documentation

4.17.3.1 **typedef struct event EVENT**

Event structure

4.17.3.2 **typedef struct event_list EVENT_LIST**

Event list structure

4.17.3.3 **typedef struct event_info EVENTINFO**

Information in an event

4.17.4 Function Documentation

4.17.4.1 **int event_init(EVENT * e)**

Initialize parameters in Event structure

Parameters

<i>e</i>	: Event
----------	---------

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.17.4.2 int event_list_get_first (EVENT_LIST * el, EVENT ** e)

Get the first event in the event-list

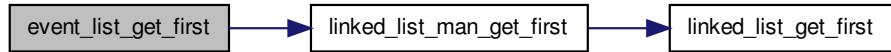
Parameters

<i>el</i>	: Event-list
<i>e</i>	: Output -> Event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.17.4.3 int event_list_init (EVENT_LIST * el)

Initialize parametters in EVENT_LIST

Parameters

<i>el</i>	: pointer to EVENT_LIST
-----------	-------------------------

Returns

Error-code (normal SUCCESS)

Here is the call graph for this function:



4.17.4.4 int event_list_insert_event(EVENT_LIST * el, EVENT * e)

Insert an event into event-list

Parameters

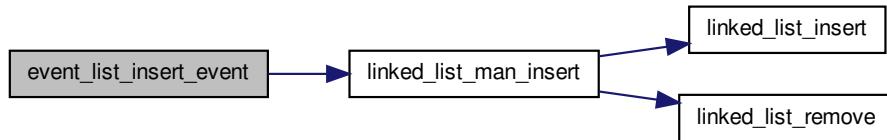
<i>el</i>	: Event list
<i>e</i>	: event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

sort from end of list, assume that [event_list](#) already sorted before

Here is the call graph for this function:



4.17.4.5 int event_list_is_empty(EVENT_LIST * l)

Check an Event-list empty or not

Parameters

<i>l</i>	: event list
----------	--------------

Returns

Return negative number if error. Return 1 if event list is empty, and return 0 otherwise.

4.17.4.6 int event_list_new_event(EVENT_LIST * el, EVENT ** e)

New event structure is allocated and init But this new event is not inserted into referenced Event-list

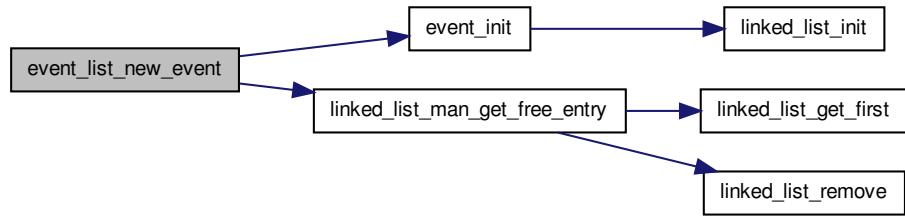
Parameters

<i>el</i>	: Event-list used for allocating new memory to Event structure
<i>e</i>	: output -> New event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:

**4.17.4.7 int event_list_remove_event (EVENT_LIST * el, EVENT * e)**

Remove an event out of event list. Note that, based on Linked-list-manager (LINKED_LIST_MAN), this event may not be freed but holding for future use (new-event).

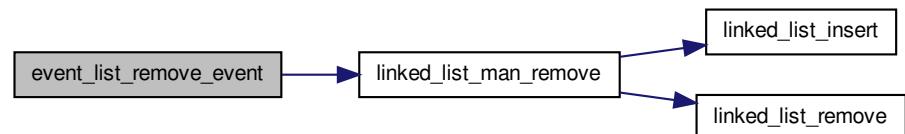
Parameters

<code>el</code>	: Event list
<code>e</code>	: removed event

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.17.4.8 int event_save (EVENT * e, FILE * f)

Save an event information into file

Parameters

<i>e</i>	: Event
<i>f</i>	: File pointer

Returns

Error code (see more in file [def.h](#) and [libs/error.h](#))

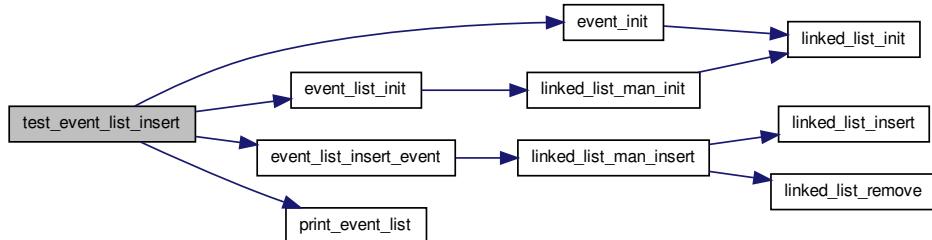
4.17.4.9 int test_event_list_insert()

Unit test of function `event_list_insert`

Returns

Error-code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:

**4.18 netsim/netsim.c File Reference**

```

#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <math.h>
#include "error.h"
#include "queues/fifo.h"
#include "conf/parser.h"
  
```

```
#include "conf/config.h"
#include "sys_aqueue.h"
#include "csma.h"
#include "netsim.h"
```

Functions

- int **event_setup** (EVENT **e*, RANDOM_CONF **fc*, TIME *curr_time*)
- int **pisas_sched** (CONFIG **conf*, SYS_STATE_OPS **sys_ops*)
- int **netsim_start** (char **conf_file*)

Variables

- CONFIG **conf**
- long **debug**

4.18.1 Detailed Description

Network simulator - the main framework.

Date

Created on: Apr 6, 2011

Author

iizke

4.18.2 Function Documentation

4.18.2.1 int event_setup (EVENT * *e*, RANDOM_CONF * *fc*, TIME *curr_time*)

Setup time of event based on its statistical characteristics.

Parameters

<i>e</i>	: Event
<i>fc</i>	: random configuration
<i>curr_time</i>	: Current time

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.18.2.2 int netsim_start (char * *conf_file*)

Main program. Do parse user configuration file, and run a chosen simulation

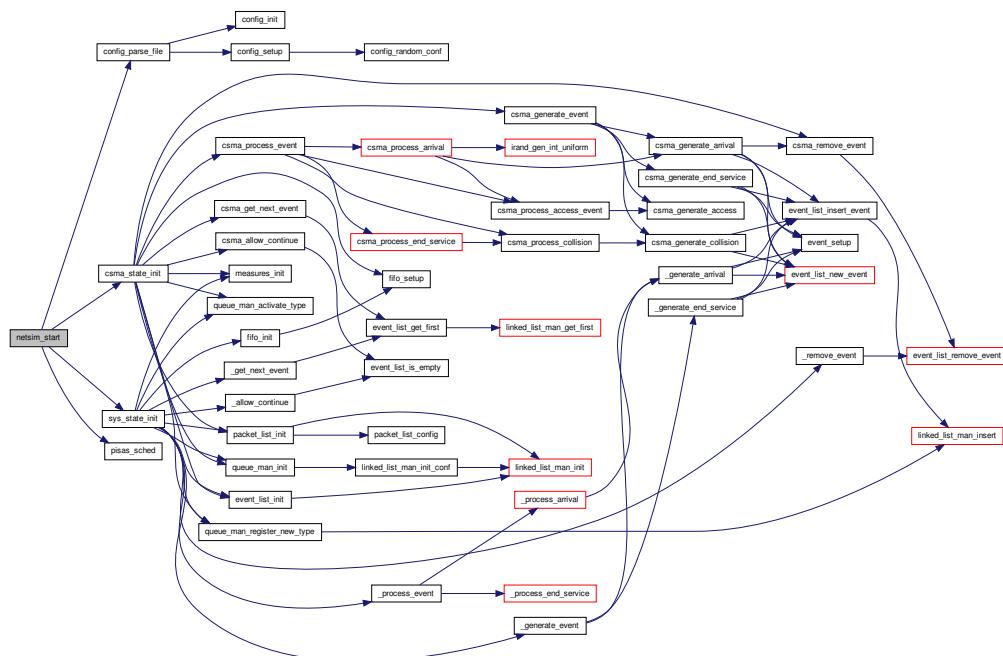
Parameters

<i>nargs</i>	: number of parameters
<i>args</i>	: parameters

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.18.2.3 int pisas_sched (CONFIG * *conf*, SYS_STATE_OPS * *sys_ops*)

Scheduling events while simulating.

Parameters

<i>conf</i>	: Configuration from user, including: arrival distribution, execution distribution, waiting line, ...
<i>sys_ops</i>	: Operations of System state.

Returns

Error code (defined in libs/error.h and [def.h](#))

4.19 netsim/netsim.h File Reference

```
#include "event.h"
#include "queues/measures.h"
#include "conf/config.h"
```

Data Structures

- struct [system_state_operations](#)
Functions of a simulated system.

Typedefs

- typedef struct [system_state_operations](#) [SYS_STATE_OPS](#)
Functions of a simulated system.

Functions

- int [event_setup](#) ([EVENT](#) **e*, [RANDOM_CONF](#) **fc*, [TIME](#) *curr_time*)
- int [netsim_start](#) (char **conf_file*)

4.19.1 Detailed Description**Date**

Created on: Apr 6, 2011

Author

iizke

4.19.2 Function Documentation**4.19.2.1 int event_setup (EVENT * *e*, RANDOM_CONF * *fc*, TIME *curr_time*)**

Setup time of event based on its statistical characteristics.

Parameters

<i>e</i>	: Event
----------	---------

<i>fc</i>	: random configuration
<i>curr_time</i>	: Current time

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.19.2.2 int netsim_start (char * *conf_file*)

Main program. Do parse user configuration file, and run a chosen simulation

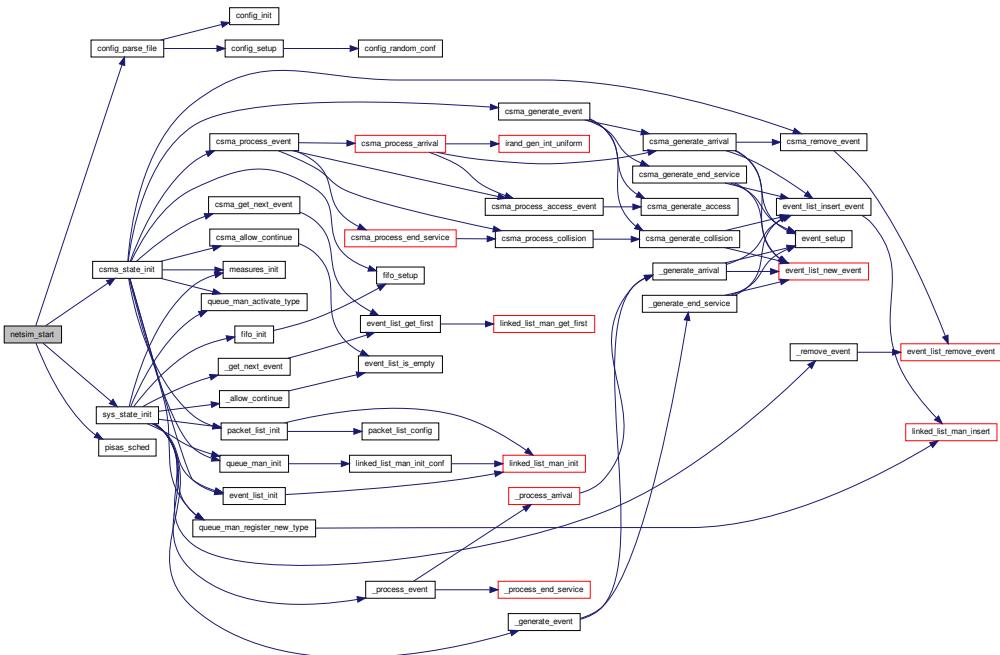
Parameters

<i>nargs</i>	: number of parameters
<i>args</i>	: parameters

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.20 netsim/queues/fifo.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "error.h"
#include "fifo.h"
```

Functions

- int [fifo_init](#) (QUEUE_TYPE **q_fifo, int max_executing, int max_waiting)
- int [fifo_setup](#) (QUEUE_TYPE *q_fifo, int max_executing, int max_waiting)

4.20.1 Detailed Description

FIFO queue implementation.

Date

Created on: Apr 16, 2011

Author

iizke

4.20.2 Function Documentation

4.20.2.1 int fifo_init (QUEUE_TYPE ** q_fifo, int max_executing, int max_waiting)

Initialization of FIFO queue, maybe allocate new memory if needed

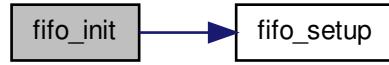
Parameters

<i>q_fifo</i>	: A queue type based on FIFO queue
<i>max_executing,:</i>	Maximum number of executing packets (negative value ~ infinite)
<i>max_waiting,:</i>	Maximum number of waiting packets (negative value ~ infinite)

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.20.2.2 int fifo_setup (QUEUE_TYPE * q_fifo, int max_executing, int max_waiting)

Setup parameters of a FIFO queue

Parameters

<i>q_fifo</i>	: FIFO queue type
<i>max_executing,:</i>	Maximum number of executing packets (negative value \sim infinite)
<i>max_waiting,:</i>	Maximum number of waiting packets (negative value \sim infinite)

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.21 netsim/queues/fifo.h File Reference

```
#include "queue_man.h"
#include "measures.h"
```

Data Structures

- struct [fifo_queue](#)

TypeDefs

- typedef struct [fifo_queue](#) FIFO_QINFO

Functions

- int [fifo_init](#) (QUEUE_TYPE **q, int max_executing, int max_waiting)
- int [fifo_setup](#) (QUEUE_TYPE *q, int max_executing, int max_waiting)

4.21.1 Detailed Description

FIFO queue structure

Date

Created on: Apr 16, 2011

Author

iizke

4.21.2 Typedef Documentation

4.21.2.1 `typedef struct fifo_queue FIFO_QINFO`

FIFO queue structure

4.21.3 Function Documentation

4.21.3.1 `int fifo_init(QUEUE_TYPE **q_fifo, int max_executing, int max_waiting)`

Initialization of FIFO queue, maybe allocate new memory if needed

Parameters

<code>q_fifo</code>	: A queue type based on FIFO queue
<code>max_executing,:</code>	Maximum number of executing packets (negative value ~ infinite)
<code>max_waiting,:</code>	Maximum number of waiting packets (negative value ~ infinite)

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.21.3.2 int fifo_setup (QUEUE_TYPE * *q_fifo*, int *max_executing*, int *max_waiting*)

Setup parameters of a FIFO queue

Parameters

<i>q_fifo</i>	: FIFO queue type
<i>max_executing</i> :	Maximum number of executing packets (negative value ~ infinite)
<i>max_waiting</i> :	Maximum number of waiting packets (negative value ~ infinite)

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.22 netsim/queues/measures.c File Reference

```
#include <string.h>
#include <stdio.h>
#include "error.h"
#include "packet.h"
#include "measures.h"
#include "queue_man.h"
```

Defines

- #define **print_statistical_value**(*_var_name*, *_var*, *_conf*)

Functions

- int **measures_init** (MEASURES **m*)
- int **print_measurement** (MEASURES **m*)
- int **measurement_collect_data** (MEASURES **m*, PACKET **p*, TIME *curr_time*)

4.22.1 Detailed Description

Measurement functions

Date

Created on: Apr 9, 2011

Author

iizke

4.22.2 Define Documentation

4.22.2.1 `#define print_statistical_value(_var_name, _var, _conf)`

Value:

```
{ \
    printf("%20s : mean %4.5f, var %4.5f, min %4.3f, max %4.3f, confidency %4.3f\
n", \
        _var_name, \
        (_var)->avg, \
        (_var)->var, \
        (_var)->min, \
        (_var)->max, \
        stat_num_calc_confidence_interval(_var, _conf)); }
```

4.22.3 Function Documentation

4.22.3.1 `int measurement_collect_data (MEASURES * m, PACKET * p, TIME curr_time)`

Collect new data into measurement structure

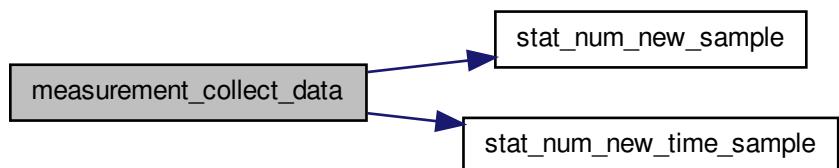
Parameters

<i>m</i>	: measurement
<i>p</i>	: packet
<i>curr_time</i>	: Current time

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.22.3.2 int measures_init (MEASURES * m)

Initialization of structure MEASURES

Parameters

<i>m</i>	: pointer to Measures structure
----------	---------------------------------

Returns

Error code (defined in [def.h](#))

4.22.3.3 int print_measurement (MEASURES * m)

Print out to screen measured information

Parameters

<i>m</i>	: pointer to a MEASURES structure.
----------	------------------------------------

Returns

Error code (defined in [def.h](#))

4.23 netsim/queues/measures.h File Reference

```
#include "def.h"
#include "stat_num.h"
#include "stat_num.h"
```

Data Structures

- struct [measures](#)

Typedefs

- typedef struct [measures](#) MEASURES

Functions

- int [measures_init](#) (MEASURES *m)
- int [print_measurement](#) (MEASURES *m)
- int [measurement_collect_data](#) (MEASURES *m, PACKET *p, TIME curr_time)

4.23.1 Detailed Description

Collect statistical information of queueing system

Date

Created on: Apr 6, 2011

Author

iizke

4.23.2 Typedef Documentation

4.23.2.1 **typedef struct measures MEASURES**

MEASURE structure used to store some results when simulating queue system

4.23.3 Function Documentation

4.23.3.1 **int measurement_collect_data (MEASURES * m, PACKET * p, TIME curr_time)**

Collect new data into measurement structure

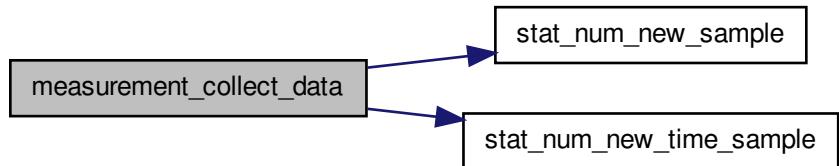
Parameters

<i>m</i>	: measurement
<i>p</i>	: packet
<i>curr_time</i>	: Current time

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.23.3.2 int measures_init (MEASURES * m)

Initialization of structure MEASURES

Parameters

<i>m</i>	: pointer to Measures structure
----------	---------------------------------

Returns

Error code (defined in [def.h](#))

4.23.3.3 int print_measurement (MEASURES * m)

Print out to screen measured information

Parameters

<i>m</i>	: pointer to a MEASURES structure.
----------	------------------------------------

Returns

Error code (defined in [def.h](#))

4.24 netsim/queues/packet.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "error.h"
#include "packet.h"
#include "fifo.h"
```

Functions

- [int packet_init \(PACKET *p\)](#)
- [int packet_list_init \(PACKET_LIST *l, int conf\)](#)
- [int packet_list_new_packet \(PACKET_LIST *pf, PACKET **p\)](#)
- [int packet_list_insert_packet \(PACKET_LIST *pf, PACKET *p\)](#)
- [int packet_list_remove_packet \(PACKET_LIST *pf, PACKET *p\)](#)
- [int packet_list_get_first \(PACKET_LIST *pf, PACKET **p\)](#)
- [int packet_list_is_empty \(PACKET_LIST *l\)](#)
- [int packet_list_config \(PACKET_LIST *pl, int conf\)](#)
- [int test_packet_list_new_packet \(\)](#)
- [int measurement_self_collect_data \(PACKET *p\)](#)

4.24.1 Detailed Description

Packet functions

Date

Created on: Apr 8, 2011

Author

iizke

4.24.2 Function Documentation

4.24.2.1 int measurement_self_collect_data (PACKET * p)

Collect data (packet) for measurement (from appropriate queue containing this packet)

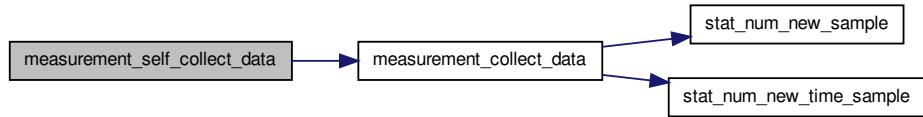
Parameters

<i>p</i>	: Packet
----------	----------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.24.2.2 int packet_init (PACKET * p)

Initialization a packet

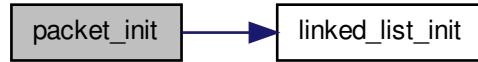
Parameters

<i>p</i>	: Packet
----------	----------

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.24.2.3 int packet_list_config (PACKET_LIST * pl, int conf)

Configure packet list

Parameters

<i>pl</i>	: Packet list
<i>conf</i>	: see libs/list/linked_list.h for more info about this configuration

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

4.24.2.4 int packet_list_get_first (PACKET_LIST * pf, PACKET ** p)

Get one packet from packet list.

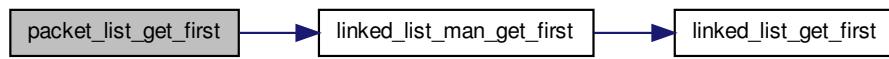
Parameters

<i>pf</i>	: packet list
<i>p</i>	: packet

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.24.2.5 int packet_list_init (PACKET_LIST * *l*, int *conf*)

Initialization of a packet list

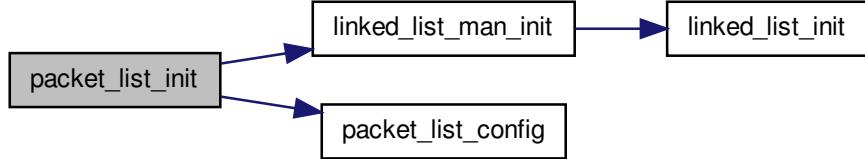
Parameters

<i>l</i>	: Packet list
<i>conf</i>	: configuration of list

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.24.2.6 int packet_list_insert_packet (PACKET_LIST * *pf*, PACKET * *p*)

Insert a packet into packet-list

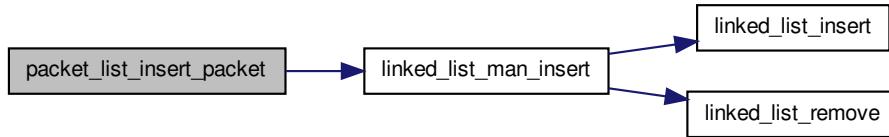
Parameters

<i>pf</i>	: packet list
<i>p</i>	: packet

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.24.2.7 int packet_list_is_empty (PACKET_LIST * l)

Check packet-list empty or not.

Parameters

<i>l</i>	: packet list
----------	---------------

Returns

negative number if there are some errors. Return 1 if list is empty, otherwise return 0.

4.24.2.8 int packet_list_new_packet (PACKET_LIST * pf, PACKET ** p)

Create new packet. If there is no free packet in packet-list, new memory is allocated to new packet. Note: if packet list is configured to no used free-packets, always allocate new memory to packet.

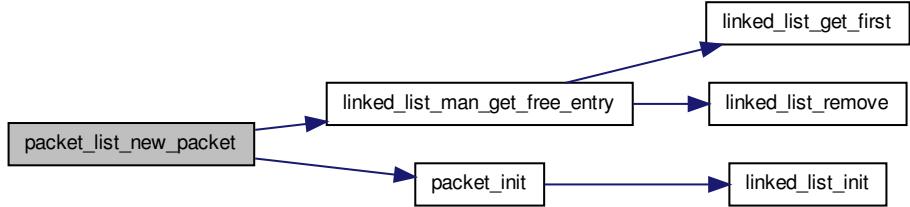
Parameters

<i>pf</i>	: packet list
<i>p</i>	: new packet (output)

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.24.2.9 int packet_list_remove_packet(PACKET_LIST * pf, PACKET * p)

Remove one packet out of packet list.

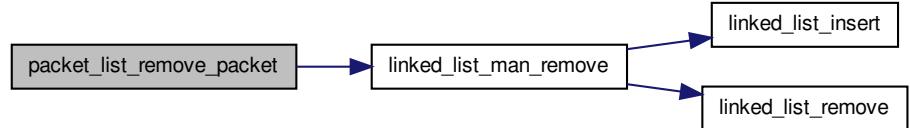
Parameters

<code>pf</code>	: packet list
<code>p</code>	: packet

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



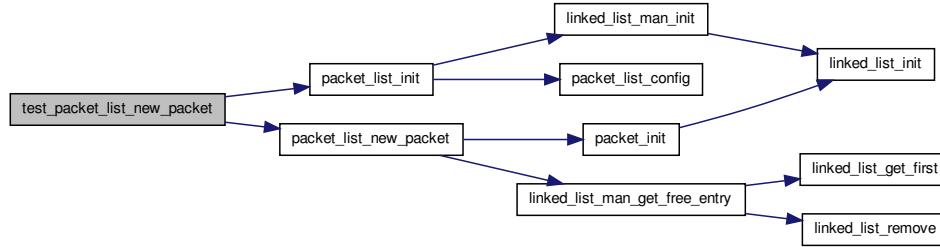
4.24.2.10 int test_packet_list_new_packet()

Unit test of function `packet_list_new_packet`

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.25 netsim/queues/packet.h File Reference

```
#include "def.h"
#include "list/linked_list.h"
```

Data Structures

- struct `packet_info`
- struct `packet`
- struct `packet_list`

Defines

- #define `PACKET_STATE_IN` 1
Packet state IN: comming to queue.
- #define `PACKET_STATE_OUT` 2
Packet goes out of queue.
- #define `PACKET_STATE_PROCESSING` 3
Packet is going to be processed.
- #define `PACKET_STATE_WAITING` 4
Packet in waiting list.
- #define `PACKET_STATE_DROPPED` 5
Packet is dropped.

TypeDefs

- `typedef struct queue_type QUEUE_TYPE`
An alias of struct queue_type.
- `typedef struct packet_info PACKET_INFO`
- `typedef struct packet PACKET`
- `typedef struct packet_list PACKET_LIST`

Functions

- `int packet_init (PACKET *p)`
- `int packet_list_init (PACKET_LIST *el, int conf)`
- `int packet_list_new_packet (PACKET_LIST *el, PACKET **e)`
- `int packet_list_insert_packet (PACKET_LIST *el, PACKET *e)`
- `int packet_list_remove_packet (PACKET_LIST *el, PACKET *e)`
- `int packet_list_get_first (PACKET_LIST *el, PACKET **e)`
- `int packet_list_is_empty (PACKET_LIST *l)`
- `int packet_list_config (PACKET_LIST *el, int conf)`
- `int test_packet_list_new_packet ()`
- `int measurement_self_collect_data (PACKET *p)`

4.25.1 Detailed Description

Packet structure

Date

Created on: Apr 8, 2011

Author

iizke

4.25.2 Typedef Documentation

4.25.2.1 `typedef struct packet PACKET`

Packet structure

4.25.2.2 `typedef struct packet_info PACKET_INFO`

Packet information

4.25.2.3 `typedef struct packet_list PACKET_LIST`

Packet list structure

4.25.3 Function Documentation

4.25.3.1 int measurement_self_collect_data(PACKET * p)

Collect data (packet) for measurement (from appropriate queue containing this packet

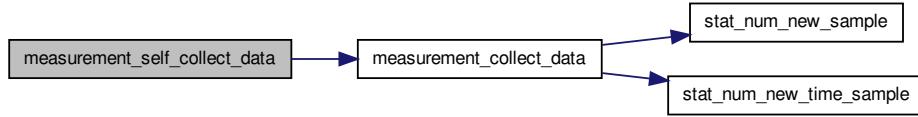
Parameters

<i>p</i>	: Packet
----------	----------

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.25.3.2 int packet_init(PACKET * p)

Initialization a packet

Parameters

<i>p</i>	: Packet
----------	----------

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.25.3.3 int packet_list_config (PACKET_LIST * *pl*, int *conf*)

Configure packet list

Parameters

<i>pl</i>	: Packet list
<i>conf</i>	: see libs/list/linked_list.h for more info about this configuration

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

4.25.3.4 int packet_list_get_first (PACKET_LIST * *pf*, PACKET ** *p*)

Get one packet from packet list.

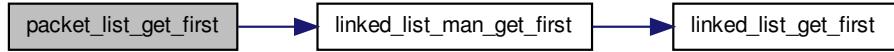
Parameters

<i>pf</i>	: packet list
<i>p</i>	: packet

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:

**4.25.3.5 int packet_list_init (PACKET_LIST * *l*, int *conf*)**

Initialization of a packet list

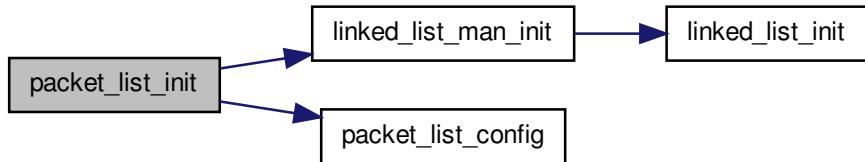
Parameters

<i>l</i>	: Packet list
<i>conf</i>	: configuration of list

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.25.3.6 int packet_list.insert_packet (PACKET_LIST * *pf*, PACKET * *p*)

Insert a packet into packet-list

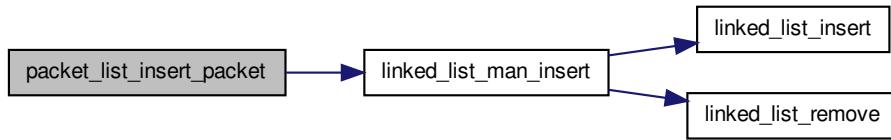
Parameters

<i>pf</i>	: packet list
<i>p</i>	: packet

Returns

Error code (defined in `def.h` and `libs/error.h`)

Here is the call graph for this function:



4.25.3.7 int packet_list.is_empty (PACKET_LIST * *l*)

Check packet-list empty or not.

Parameters

<i>l</i>	: packet list
----------	---------------

Returns

negative number if there are some errors. Return 1 if list is empty, otherwise return 0.

4.25.3.8 int packet_list_new_packet (PACKET_LIST * *pf*, PACKET ** *p*)

Create new packet. If there is no free packet in packet-list, new memory is allocated to new packet. Note: if packet list is configured to no used free-packets, always allocate new memory to packet.

Parameters

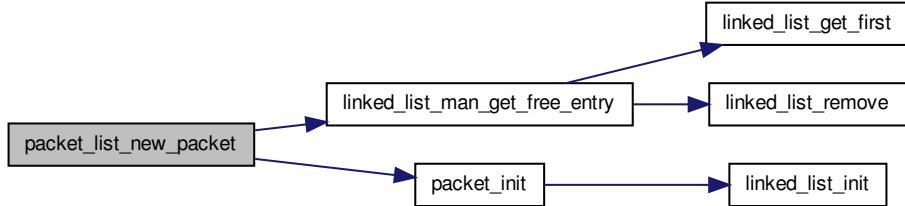
<i>pf</i>	: packet list
-----------	---------------

<i>p</i>	: new packet (output)
----------	-----------------------

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:

**4.25.3.9 int packet_list_remove_packet (PACKET_LIST * *pf*, PACKET * *p*)**

Remove one packet out of packet list.

Parameters

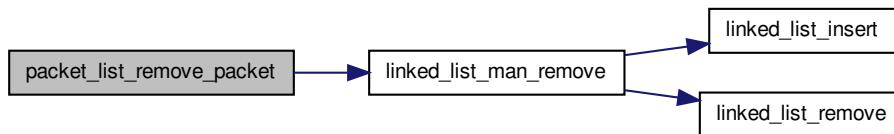
<i>pf</i>	: packet list
-----------	---------------

<i>p</i>	: packet
----------	----------

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:

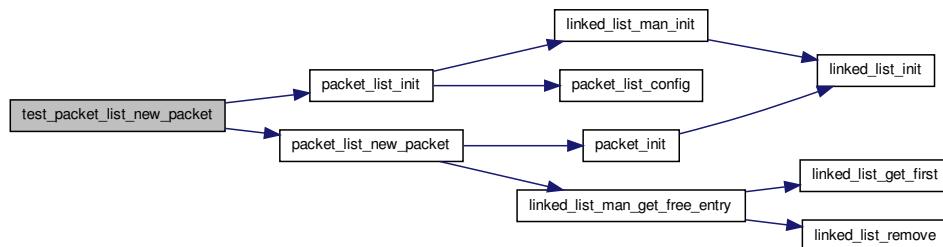
**4.25.3.10 int test_packet_list_new_packet()**

Unit test of function `packet_list_new_packet`

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:

**4.26 netsim/queues/queue_man.c File Reference**

```

#include <stdio.h>
#include "queue_man.h"
#include "error.h"
  
```

Functions

- int `queue_man_init` (QUEUE_MAN *qm)
- int `queue_man_register_new_type` (QUEUE_MAN *qm, QUEUE_TYPE *qi)
- int `queue_man_unregister_type` (QUEUE_MAN *qm, QUEUE_TYPE *qi)
- int `queue_man_activate_type` (QUEUE_MAN *qm, int type)

4.26.1 Detailed Description

Implementation of essential queue operations.

Date

Created on: Apr 16, 2011

Author

iizke

4.26.2 Function Documentation

4.26.2.1 int queue_man_activate_type (QUEUE_MAN * *qm*, int *type*)

Activate a queue-type in queue-manager. Simulation is based on this queue-type.

Parameters

<i>qm</i>	: queue manager
<i>type</i>	: queue type

Returns

Error code (defined in `def.h` and `libs/error.h`)

4.26.2.2 int queue_man_init (QUEUE_MAN * *qm*)

Initialization of a Queue manager

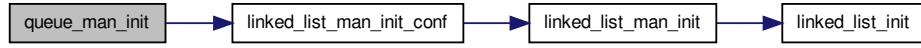
Parameters

<i>qm</i>	: queue manager
-----------	-----------------

Returns

Error code (defined in `def.h` and `libs/error.h`)

Here is the call graph for this function:



4.26.2.3 int queue_man_register_new_type (QUEUE_MAN * *qm*, QUEUE_TYPE * *qi*)

Register new queue-type into queue-manager

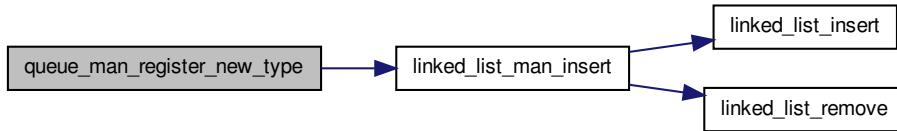
Parameters

<i>qm</i>	: queue manager
<i>qi</i>	: queue type

Returns

Error code (defined in [def.h](#) and libs/error.h)

Here is the call graph for this function:



4.26.2.4 int queue_man_unregister_type (QUEUE_MAN * *qm*, QUEUE_TYPE * *qi*)

Remove a queue type out of queue-manager

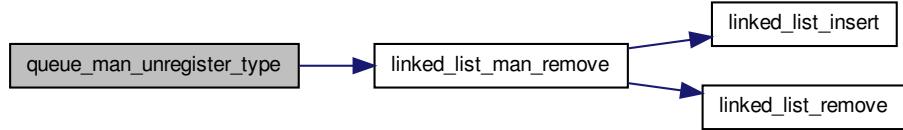
Parameters

<i>qm</i>	: queue manager
<i>qi</i>	: queue type

Returns

Error code (defined in [def.h](#) and libs/error.h)

Here is the call graph for this function:



4.27 netsim/queues/queue_man.h File Reference

```
#include "packet.h"
```

Data Structures

- struct [queue_type](#)
- struct [queue_type_list](#)

Defines

- #define [QUEUE_FIFO](#) 1
Queue type FIFO.

Typedefs

- typedef struct [queue_type_list](#) [QUEUE_MAN](#)

Functions

- int [queue_man_init](#) ([QUEUE_MAN](#) *qm)
- int [queue_man_register_new_type](#) ([QUEUE_MAN](#) *qm, [QUEUE_TYPE](#) *qi)
- int [queue_man_unregister_type](#) ([QUEUE_MAN](#) *qm, [QUEUE_TYPE](#) *qi)
- int [queue_man_activate_type](#) ([QUEUE_MAN](#) *qm, int type)

4.27.1 Detailed Description

Manage all types of queue, for example: FIFO, LIFO, RR,... At this time, only FIFO is implemented.

Date

Created on: Apr 16, 2011

Author

iizke

4.27.2 Typedef Documentation**4.27.2.1 `typedef struct queue_type_list QUEUE_MAN`**

Queue manager structure

4.27.3 Function Documentation**4.27.3.1 `int queue_man_activate_type (QUEUE_MAN * qm, int type)`**

Activate a queue-type in queue-manager. Simulation is based on this queue-type.

Parameters

<code>qm</code>	: queue manager
<code>type</code>	: queue type

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

4.27.3.2 `int queue_man_init (QUEUE_MAN * qm)`

Initialization of a Queue manager

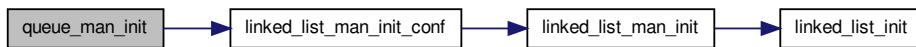
Parameters

<code>qm</code>	: queue manager
-----------------	-----------------

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.27.3.3 int queue_man_register_new_type (QUEUE_MAN * *qm*, QUEUE_TYPE * *qi*)

Register new queue-type into queue-manager

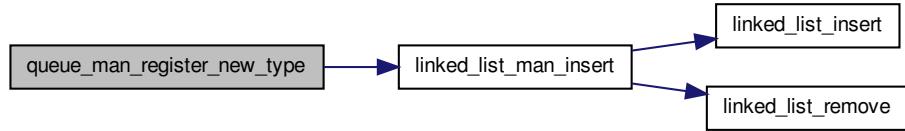
Parameters

<i>qm</i>	: queue manager
<i>qi</i>	: queue type

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.27.3.4 int queue_man_unregister_type (QUEUE_MAN * *qm*, QUEUE_TYPE * *qi*)

Remove a queue type out of queue-manager

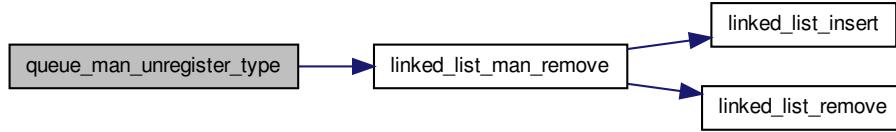
Parameters

<i>qm</i>	: queue manager
<i>qi</i>	: queue type

Returns

Error code (defined in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.28 netsim/sys_aqueue.c File Reference

```
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include "error.h"
#include "queues/fifo.h"
#include "sys_aqueue.h"
```

Functions

- EVENT * _generate_arrival (CONFIG *conf, SYS_STATE *state)
- EVENT * _generate_end_service (PACKET *p, CONFIG *conf, SYS_STATE *state)
- int _allow_continue (CONFIG *conf, SYS_STATE_OPS *ops)
- int _packet_from_event (EVENT *e, PACKET *p)
- int _process_arrival (EVENT *e, CONFIG *conf, SYS_STATE *state)
- int _process_end_service (EVENT *e, CONFIG *conf, SYS_STATE *state)
- EVENT * _generate_event (int type, PACKET *p, CONFIG *conf, SYS_STATE_OPS *ops)
- int _process_event (EVENT *e, CONFIG *conf, SYS_STATE_OPS *ops)
- int _get_next_event (SYS_STATE_OPS *ops, EVENT **e)
- int _remove_event (SYS_STATE_OPS *ops, EVENT *e)
- int sys_state_init (SYS_STATE *state, CONFIG *conf)

4.28.1 Detailed Description

Simulation of system with one queue (but can be multi-servers)

Date

Created on: May 19, 2011

Author

iizke

4.28.2 Function Documentation**4.28.2.1 int _allow_continue(CONFIG * *conf*, SYS_STATE_OPS * *ops*)**

Check whether system should be stopped or not.

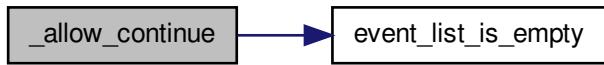
Parameters

<i>conf</i>	: User configuration
<i>ops</i>	: Abstract system operators

Returns

0 if system should be stopped, 1 otherwise.

Here is the call graph for this function:

**4.28.2.2 EVENT* _generate_arrival(CONFIG * *conf*, SYS_STATE * *state*)**

Generate arrival event

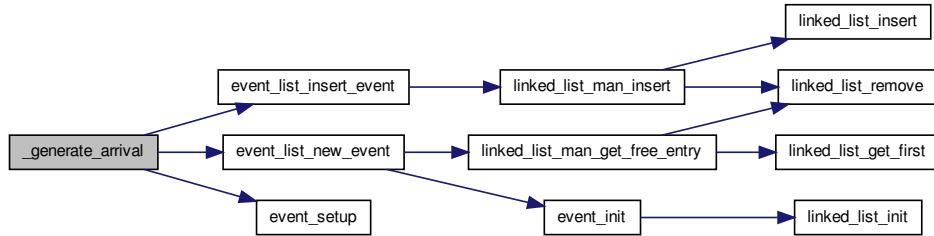
Parameters

<i>conf</i>	: user configuration
<i>state</i>	: system state

Returns

new event (already added in Event list)

Here is the call graph for this function:



4.28.2.3 EVENT* _generate_end_service (PACKET * *p*, CONFIG * *conf*, SYS_STATE * *state*)

Generate new end-service event.

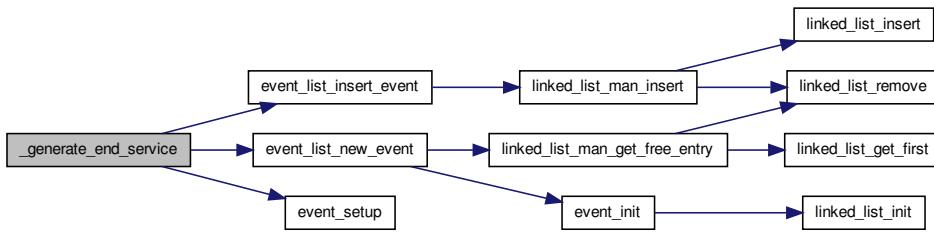
Parameters

<i>p</i>	: Processing packet (for this event)
<i>conf</i>	: user configuration
<i>state</i>	: System state

Returns

New event (already added in the event list)

Here is the call graph for this function:



4.28.2.4 EVENT* `_generate_event(int type, PACKET * p, CONFIG * conf, SYS_STATE_OPS * ops)`

Generate new event.

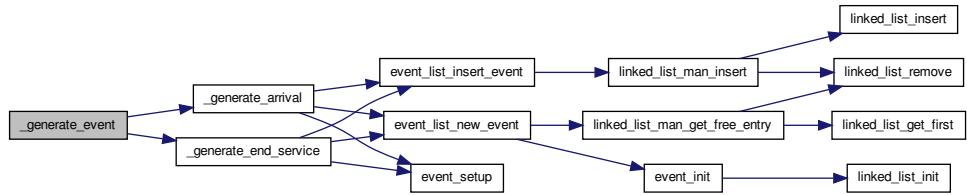
Parameters

<code>type</code>	: type of event
<code>p</code>	: Packet related to this new event
<code>conf</code>	: user configuration
<code>ops</code>	: Abstract system operations

Returns

New event

Here is the call graph for this function:



4.28.2.5 int `_get_next_event(SYS_STATE_OPS * ops, EVENT ** e)`

Get next event from event list

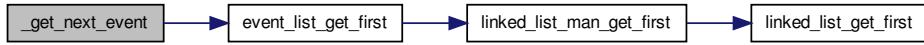
Parameters

<code>ops</code>	: Abstract system operations
<code>e</code>	: returned event

Returns

Error code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.28.2.6 int _packet_from_event(EVENT * e, PACKET * p)

Setup time and type of packet extracted from an event

Parameters

<i>e</i>	: Event
<i>p</i>	: Packet

Returns

Error code (see more in [def.h](#) and [error.h](#))

4.28.2.7 int _process_arrival(EVENT * e, CONFIG * conf, SYS_STATE * state)

Process Arrival event

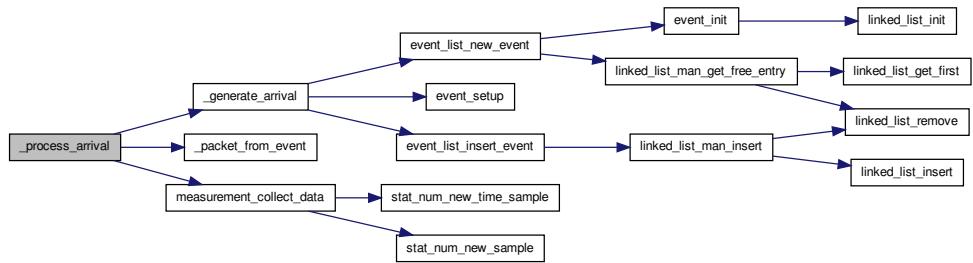
Parameters

<i>e</i>	: Event
<i>conf</i>	: configuration from user
<i>state</i>	: system state

Returns

Error code (see more in [def.h](#) and [libs/error.h](#))

Here is the call graph for this function:



4.28.2.8 int _process_end_service (EVENT * e, CONFIG * conf, SYS_STATE * state)

Process end-service event

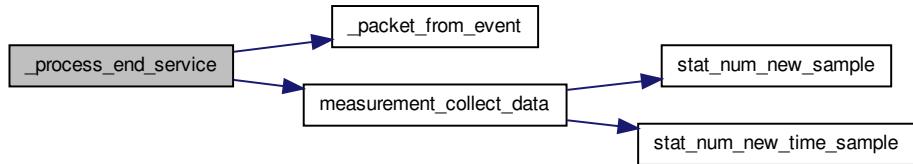
Parameters

<i>e</i>	: event
<i>conf</i>	: user configuration
<i>state</i>	: system state

Returns

Error code (see more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.28.2.9 int _process_event (EVENT * e, CONFIG * conf, SYS_STATE_OPS * ops)

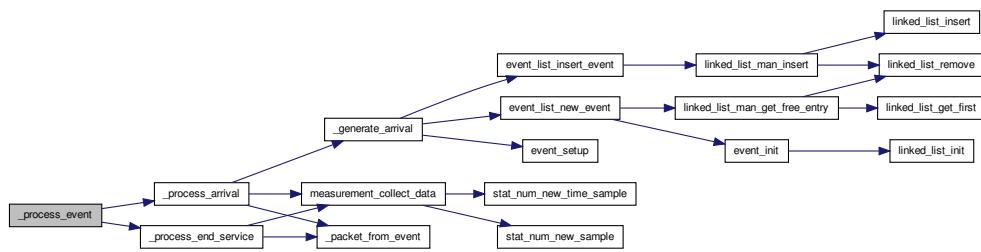
Process an event.

Parameters

<i>e</i>	: Event
<i>conf</i>	: User configuration
<i>ops</i>	: Abstract system operations

ReturnsError code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:

**4.28.2.10 int _remove_event(SYS_STATE_OPS * ops, EVENT * e)**

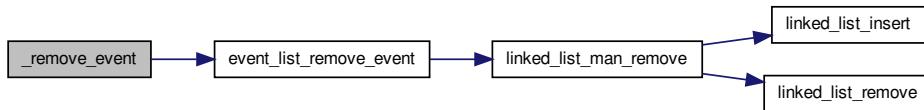
Remove an event out of event list

Parameters

<i>ops</i>	: Abstract system operations
<i>e</i>	: Event

ReturnsError code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.28.2.11 int sys_state_init (SYS_STATE * state, CONFIG * conf)

Initialize system state of one-queue system

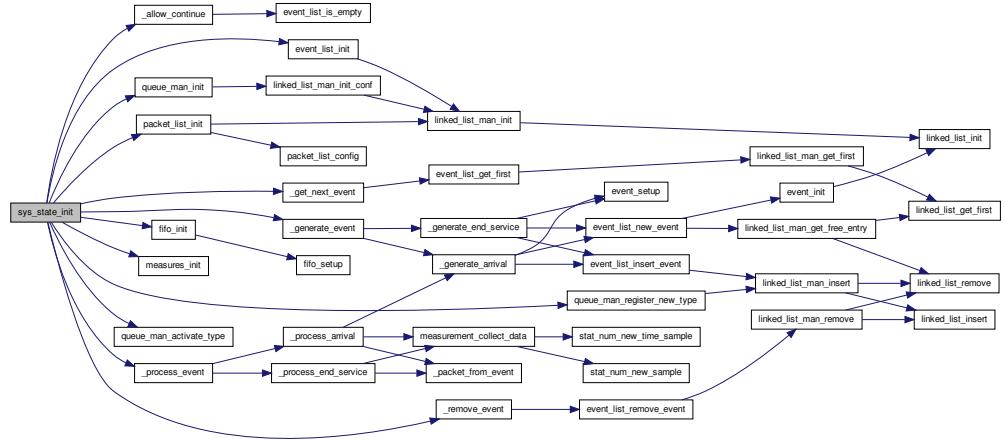
Parameters

<i>state</i>	: system state
<i>conf</i>	: user configuration

Returns

Error code (more in [def.h](#) and [error.h](#))

Here is the call graph for this function:



4.29 netsim/sys_aqueue.h File Reference

```

#include "queues/measures.h"
#include "queues/queue_man.h"
#include "event.h"
#include "netsim.h"
  
```

Data Structures

- struct [sys_state](#)

Defines

- #define `get_sys_state_from_ops(_ops)` (container_of(_ops, SYS_STATE, ops))

Typedefs

- typedef struct `sys_state` SYS_STATE

Functions

- int `sys_state_init (SYS_STATE *state, CONFIG *conf)`

4.29.1 Detailed Description

Specification of state in system with one queue

Date

Created on: May 19, 2011

Author

iizke

4.29.2 Typedef Documentation**4.29.2.1 typedef struct sys_state SYS_STATE**

Structure representing the system state in simulation.

4.29.3 Function Documentation**4.29.3.1 int sys_state_init (SYS_STATE * *state*, CONFIG * *conf*)**

Initialize system state of one-queue system

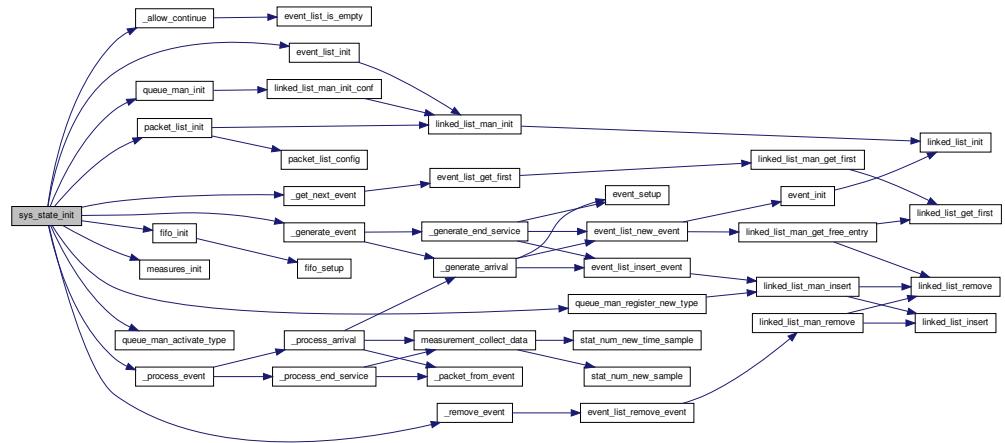
Parameters

<code>state</code>	: system state
<code>conf</code>	: user configuration

Returns

Error code (more in `def.h` and `error.h`)

Here is the call graph for this function:



4.30 network.c File Reference

```
#include <stdlib.h>
#include "error.h"
#include "network.h"
```

Functions

- int **nw_get_neighbor** (**NETWORK** *nw, int node, **NW_NODE_LIST** *neighbor)
 - int **nw_node_list_scan** (**NW_NODE_LIST** *nl, int *nodeid)
 - int **sp_distance_list_init0** (**SP_DISTANCE_LIST** **list)
 - int **sp_distance_list_init** (**SP_DISTANCE_LIST** **list, int nentries, int src)
 - int **sp_distance_list_remove_smallest** (**SP_DISTANCE_LIST** *l, **SP_DISTANCE** **d)
 - int **sp_distance_list_insert** (**SP_DISTANCE_LIST** *l, **SP_DISTANCE** *d)
 - **SP_DISTANCE** * **sp_distance_list_get** (**SP_DISTANCE_LIST** *l, int id)
 - float **sp_distance_list_get_value** (**SP_DISTANCE_LIST** *l, int id)
 - int **sp_distance_list_update** (**SP_DISTANCE_LIST** *l, int end_node, int last_node, float d)
 - int **sp_distance_list_is_empty** (**SP_DISTANCE_LIST** *l)
 - **SP_DISTANCE_LIST** * **_nw_shortest_path_dijkstra** (**NETWORK** *net, int src)
 - void * **nw_find_shortest_path** (**NETWORK** *net, int alg, int src, int dest)

4.30.1 Detailed Description

Networking problem of operation research: For example Shortest path, Min Cost Flow,
...

Date

Created on: May 25, 2011

Author

iizke

4.31 network.h File Reference

```
#include "list/linked_list.h"
#include "matrix/matrix.h"
```

Data Structures

- struct `shortest_path_distance`
- struct `sp_distance_list`
- struct `network`
- struct `network_node_list`

Defines

- #define `NETWORK_SP_DIJKSTRA` 1
- #define `NETWORK_SP_BELLFORD` 2
- #define `nw_get_cost`(nw, r, c) `matrix_get_value(&nw->costs, r, c)`

Typedefs

- typedef struct `shortest_path_distance` `SP_DISTANCE`
- typedef struct `sp_distance_list` `SP_DISTANCE_LIST`
- typedef struct `network` `NETWORK`
- typedef struct `network_node_list` `NW_NODE_LIST`

Functions

- void * `nw_find_shortest_path` (`NETWORK` *, int, int, int)
- int `nw_get_neighbor` (`NETWORK` *, int, `NW_NODE_LIST` *)
- int `nw_node_list_scan` (`NW_NODE_LIST` *, int *)
- int `sp_distance_list_init` (`SP_DISTANCE_LIST` **list, int nentries, int src)

- int `sp_distance_list_remove_smallest`(`SP_DISTANCE_LIST` *, `SP_DISTANCE` **)
- int `sp_distance_list_insert`(`SP_DISTANCE_LIST` *, `SP_DISTANCE` *)
- float `sp_distance_list_get_value`(`SP_DISTANCE_LIST` *, int)
- int `sp_distance_list_update`(`SP_DISTANCE_LIST` *, int, int, float)
- int `sp_distance_list_is_empty`(`SP_DISTANCE_LIST` *)
- `SP_DISTANCE` * `sp_distance_list_get`(`SP_DISTANCE_LIST` *l, int id)

4.31.1 Detailed Description

Network structure: traffic, cost, ...

Date

Created on: May 25, 2011

Author

iizke

4.32 quantile.h File Reference

Defines

- #define `pvalue_chi_id`(`p_value`)
- #define `get_chisqr_pvalue`(`_d`, `_p`) (`chisqr_pvalues[_d - 1][pvalue_chi_id(_p)]`)
- #define `pvalue_normal_id`(`p_value`)
- #define `get_normal_pvalue`(`_p`) (`normal_pvalues[pvalue_normal_id(_p)]`)

4.32.1 Detailed Description

Support quantile of essential distributions

Date

Created on: May 26, 2011

Author

iizke

4.32.2 Define Documentation

4.32.2.1 #define pvalue_chi_id(p_value)

Value:

```
((p_value == 0.95) ? 0 : \
((p_value == 0.9) ? 1 : \
((p_value == 0.8) ? 2 : \
((p_value == 0.7) ? 3 : \
((p_value == 0.5) ? 4 : \
((p_value == 0.3) ? 5 : \
((p_value == 0.2) ? 6 : \
((p_value == 0.1) ? 7 : \
((p_value == 0.05) ? 8 : \
((p_value == 0.01) ? 9 : \
((p_value == 0.001) ? 10 : (-1)) \
))))))))
```

4.32.2.2 #define pvalue_normal_id(p_value)

Value:

```
((p_value == 0.9) ? 0 : \
((p_value == 0.95) ? 1 : \
((p_value == 0.975) ? 2 : \
((p_value == 0.99) ? 3 : \
((p_value == 0.995) ? 4 : \
((p_value == 0.999) ? 5 : (-1)) \
))))
```

4.33 stat_num.c File Reference

```
#include <string.h>
#include <stdio.h>
#include "stat_num.h"
#include "error.h"
#include "math.h"
#include "quantile.h"
```

Defines

- #define **min**(a, b) (a==0?b:(a<b?a:b))

Calculate minimum of two values.

- #define **max**(a, b) (a<b?b:a)

Calculate maximum of two value.

Functions

- int **stat_num_new_sample** (STAT_NUM *sn, float sample)

- int `stat_num_new_time_sample (STAT_NUM *sn, float sample, float time)`
- double `stat_num_calc_confidence_interval (STAT_NUM *sn, double confidence)`
- int `stat_num_init (STAT_NUM *m)`

4.33.1 Detailed Description

Statistical calculation

Date

Created on: Apr 12, 2011

Author

iizke

4.33.2 Function Documentation

4.33.2.1 int stat_num_init (STAT_NUM * m)

Initialization of STAT_NUM structure (statistical information)

Parameters

<code>m</code>	: STAT_NUM
----------------	------------

Returns

Error code (defined in [error.h](#))

4.33.2.2 int stat_num_new_sample (STAT_NUM * sn, float sample)

Calculate the statistical values of a random variable through its samples. Used for discrete random variable.

Parameters

<code>sn</code>	: Statistical info
<code>sample</code>	: new sample is collected to statistical info

Returns

Error code (defined in [error.h](#))

4.33.2.3 int stat_num_new_time_sample (STAT_NUM *sn, float sample, float time)

Calculate the statistical values of a random variable through its samples. Used for continuous random variable.

Parameters

<i>sn</i>	: statistical information
<i>sample</i>	: new sample value
<i>time</i>	: happened time for new sample

Returns

Error code (defined in [error.h](#))

4.34 stat_num.h File Reference

Data Structures

- struct [statistical_number](#)

TypeDefs

- typedef struct [statistical_number](#) STAT_NUM

Functions

- int [stat_num_new_sample](#) (STAT_NUM *snum, float sample)
- int [stat_num_new_time_sample](#) (STAT_NUM *sn, float sample, float time)
- int [stat_num_init](#) (STAT_NUM *snum)
- double [stat_num_calc_confidence_interval](#) (STAT_NUM *sn, double confidence)

4.34.1 Detailed Description

Statistical number structure

Date

Created on: Apr 12, 2011

Author

iizke

4.34.2 Typedef Documentation

4.34.2.1 `typedef struct statistical_number STAT_NUM`

Statistical information of a random process

4.34.3 Function Documentation

4.34.3.1 `int stat_num_init(STAT_NUM * m)`

Initialization of STAT_NUM structure (statistical information)

Parameters

<code>m</code>	: STAT_NUM
----------------	------------

Returns

Error code (defined in [error.h](#))

4.34.3.2 `int stat_num_new_sample(STAT_NUM * sn, float sample)`

Calculate the statistical values of a random variable through its samples. Used for discrete random variable.

Parameters

<code>sn</code>	: Statistical info
<code>sample</code>	: new sample is collected to statistical info

Returns

Error code (defined in [error.h](#))

4.34.3.3 `int stat_num_new_time_sample(STAT_NUM * sn, float sample, float time)`

Calculate the statistical values of a random variable through its samples. Used for continuous random variable.

Parameters

<code>sn</code>	: statistical information
<code>sample</code>	: new sample value
<code>time</code>	: happened time for new sample

Returns

Error code (defined in [error.h](#))

4.35 tests/chisqr.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "../error.h"
#include "../random.h"
#include "../quantile.h"
```

Data Structures

- struct [test_params](#)

Typedefs

- typedef struct [test_params](#) TEST_PARAMS

Functions

- int [test_chisqr \(TEST_PARAMS *tp\)](#)
- int [check_chisqr_pvalue \(\)](#)

4.35.1 Detailed Description

Chi-Square test

Date

Created on: May 17, 2011

Author

iizke

4.35.2 Function Documentation

4.35.2.1 int [test_chisqr \(TEST_PARAMS * tp \)](#)

Generate pseudo random samples and count them in intervals $i/n \leq \text{sample} < (i+1)/n$
===== $(\text{sample} \times n - 1) < i \leq \text{sample} \times n$

Compute and compare chi-square value

Index

_allow_continue
 sys_aqueue.c, 162
_combined_linear_gen_init
 rndnum.c, 81
_combined_linear_gen_num
 rndnum.c, 81
_combined_linear_gen_real
 rndnum.c, 82
_composition_gen_rnd
 rnddist.c, 73
_convolution_gen_rnd
 rnddist.c, 73
_field_dis_cap, 5
_generate_arrival
 sys_aqueue.c, 162
_generate_end_service
 sys_aqueue.c, 163
_generate_event
 sys_aqueue.c, 163
_get_next_event
 sys_aqueue.c, 164
_graph, 6
_inverse_bernoulli
 rnddist.c, 74
_inverse_empirical
 rnddist.c, 74
_inverse_exp
 rnddist.c, 74
_inverse_gen_rnd
 rnddist.c, 75
_inverse_geometric
 rnddist.c, 75
_inverse_int_uniform
 rnddist.c, 75
_inverse_pareto
 rnddist.c, 76
_inverse_uniform
 rnddist.c, 76
_inverse_weibull
 rnddist.c, 76
_item_heap, 7
_linear_lehmer_gen_init
 rndnum.c, 82
_linear_lehmer_gen_num
 rndnum.c, 83
_linear_lehmer_gen_real
 rndnum.c, 83
_link, 7
_link_list, 8
_node, 9
_packet_from_event
 sys_aqueue.c, 165
_path_item, 10
_process_arrival
 sys_aqueue.c, 165
_process_end_service
 sys_aqueue.c, 166
_process_event
 sys_aqueue.c, 166
_remove_event
 sys_aqueue.c, 167
check_null_pointer
 error.h, 62
chisqr.c
 test_chisqr, 177
column, 11
combined_linear_congruential_gen, 11
COMBINED_LINEAR_GEN
 rndnum.c, 81
CONFIG
 config.h, 106
config, 12
 runtime_state, 13
config.c
 config_init, 103
 config_parse_file, 103
 config_random_conf, 104
 config_setup, 104
config.h
 CONFIG, 106
 config_init, 106

config_parse_file, 107
config_random_conf, 107
config_setup, 107
QUEUE_CONF, 106
RANDOM_CONF, 106
STOP_CONF, 106
config_init
 config.c, 103
 config.h, 106
config_parse_file
 config.c, 103
 config.h, 107
config_random_conf
 config.c, 104
 config.h, 107
config_setup
 config.c, 104
 config.h, 107
container_of
 linked_list.h, 95
csma.c
 csma_allow_continue, 109
 csma_generate_access, 110
 csma_generate_arrival, 110
 csma_generate_collision, 111
 csma_generate_end_service, 111
 csma_generate_event, 112
 csma_get_next_event, 113
 csma_process_access_event, 113
 csma_process_arrival, 114
 csma_process_collision, 114
 csma_process_end_service, 115
 csma_process_event, 116
 csma_remove_event, 116
 csma_state_init, 117
csma.h
 csma_state_init, 119
csma_allow_continue
 csma.c, 109
csma_conf, 14
csma_generate_access
 csma.c, 110
csma_generate_arrival
 csma.c, 110
csma_generate_collision
 csma.c, 111
csma_generate_end_service
 csma.c, 111
csma_generate_event
 csma.c, 112
csma_get_next_event
 csma.c, 113
csma_process_access_event
 csma.c, 113
csma_process_arrival
 csma.c, 114
csma_process_collision
 csma.c, 114
csma_process_end_service
 csma.c, 115
csma_process_event
 csma.c, 116
csma_remove_event
 csma.c, 116
csma_state_init
 csma.c, 117
csma_state_init
 csma.h, 119
dense_matrix, 16
empirical_params, 16
error
 error.c, 58
 error.h, 63
error.c, 58
 error, 58
 gc_malloc, 58
 trash_clean, 59
 trash_collect_garbage, 59
error.h, 60
 check_null_pointer, 62
 error, 63
 gc_malloc, 63
 iprint, 62
 trash_clean, 63
 trash_collect_garbage, 64
 try, 62
EVENT
 event.h, 127
event, 17
event.c
 event_init, 120
 event_list_get_first, 121
 event_list_init, 121
 event_list_insert_event, 122
 event_list_is_empty, 122
 event_list_new_event, 123

event_list_remove_event, 123
 event_save, 124
 print_event_list, 124
 test_event_list_insert, 124
 event.h
 EVENT, 127
 event_init, 127
 EVENT_LIST, 127
 event_list_get_first, 127
 event_list_init, 128
 event_list_insert_event, 128
 event_list_is_empty, 129
 event_list_new_event, 129
 event_list_remove_event, 130
 event_save, 130
 EVENTINFO, 127
 swap_prev_event, 126
 test_event_list_insert, 131
 event_info, 19
 event_init
 event.c, 120
 event.h, 127
 EVENT_LIST
 event.h, 127
 event_list, 21
 event_list_get_first
 event.c, 121
 event.h, 127
 event_list_init
 event.c, 121
 event.h, 128
 event_list_insert_event
 event.c, 122
 event.h, 128
 event_list_is_empty
 event.c, 122
 event.h, 129
 event_list_new_event
 event.c, 123
 event.h, 129
 event_list_remove_event
 event.c, 123
 event.h, 130
 event_save
 event.c, 124
 event.h, 130
 event_setup
 netsim.c, 132
 netsim.h, 134
 EVENTINFO

event.h, 127
 fifo.c
 fifo_init, 136
 fifo_setup, 137
 fifo.h
 fifo_init, 138
 FIFO_QINFO, 138
 fifo_setup, 138
 fifo_init
 fifo.c, 136
 fifo.h, 138
 FIFO_QINFO
 fifo.h, 138
 fifo_queue, 22
 fifo_setup
 fifo.c, 137
 fifo.h, 138
 garbage, 25
 gc_malloc
 error.c, 58
 error.h, 63
 heap, 25
 iprint
 error.h, 62
 irand.h
 irand_gen_bernoulli, 66
 irand_gen_erlang, 66
 irand_gen_exp, 67
 irand_gen_int_uniform, 67
 irand_gen_pareto, 68
 irand_gen_random, 68
 irand_gen_random_real, 69
 irand_gen_random, 69
 irand_gen_random_real, 69
 irand_gen_uniform, 70
 irand_init, 70
 irand_new_seed, 71
 irand_random_seed, 71
 irand/irand.h, 65
 irand/rnddist.c, 72
 irand/rndnum.c, 79
 irand_gen_bernoulli
 irand.h, 66
 rnddist.c, 76
 irand_gen_erlang
 irand.h, 66

rnddist.c, 77
irand_gen_exp
 irand.h, 67
 rnddist.c, 77
irand_gen_int_uniform
 irand.h, 67
 rnddist.c, 78
irand_gen_pareto
 irand.h, 68
 rnddist.c, 78
irand_gen_random
 irand.h, 68
 rndnum.c, 83
irand_gen_random_real
 irand.h, 69
 rndnum.c, 84
irand_gen_srandom
 irand.h, 69
 rndnum.c, 84
irand_gen_srandom_real
 irand.h, 69
 rndnum.c, 85
irand_gen_uniform
 irand.h, 70
 rnddist.c, 79
irand_init
 irand.h, 70
 rndnum.c, 85
irand_new_seed
 irand.h, 71
 rndnum.c, 86
irand_random_seed
 irand.h, 71
 rndnum.c, 86

linear_congruential_gen, 25
LINEAR_LEHMER_GEN
 rndnum.c, 81
LINKED_LIST
 linked_list.h, 96
linked_list, 26
linked_list.c
 linked_list_get_first, 88
 linked_list_init, 89
 linked_list_insert, 89
 linked_list_man_alloc, 89
 linked_list_man_get_first, 90
 linked_list_man_get_free_entry, 90
 linked_list_man_init, 91
 linked_list_man_init_conf, 91
linked_list_man_insert, 92
linked_list_man_remove, 92
linked_list_remove, 93
print_list, 93
linked_list.h
 container_of, 95
 LINKED_LIST, 96
 linked_list_get_first, 96
 linked_list_init, 96
 linked_list_insert, 96
 LINKED_LIST_MAN, 96
 linked_list_man_alloc, 97
 linked_list_man_get_first, 97
 linked_list_man_get_free_entry, 98
 linked_list_man_init, 98
 linked_list_man_init_conf, 99
 linked_list_man_insert, 99
 linked_list_man_remove, 100
 linked_list_remove, 101
 print_list, 101
linked_list_get_first
 linked_list.c, 88
 linked_list.h, 96
linked_list_init
 linked_list.c, 89
 linked_list.h, 96
linked_list_insert
 linked_list.c, 89
 linked_list.h, 96
LINKED_LIST_MAN
 linked_list.h, 96
linked_list_man_alloc
 linked_list.c, 89
 linked_list.h, 97
linked_list_man_get_first
 linked_list.c, 90
 linked_list.h, 97
linked_list_man_get_free_entry
 linked_list.c, 90
 linked_list.h, 98
linked_list_man_init
 linked_list.c, 91
 linked_list.h, 98
linked_list_man_init_conf
 linked_list.c, 91
 linked_list.h, 99
linked_list_man_insert
 linked_list.c, 92
 linked_list.h, 99
linked_list_man_remove

linked_list.c, 92
 linked_list.h, 100
 linked_list_manager, 27
 linked_list_remove
 linked_list.c, 93
 linked_list.h, 101
 list/heap.c, 87
 list/heap.h, 87
 list/linked_list.c, 88
 list/linked_list.h, 94
 matrix, 28
 measurement_collect_data
 measures.c, 140
 measures.h, 142
 measurement_self_collect_data
 packet.c, 144
 packet.h, 151
 MEASURES
 measures.h, 142
 measures, 29
 measures.c
 measurement_collect_data, 140
 measures_init, 140
 print_measurement, 141
 print_statistical_value, 140
 measures.h
 measurement_collect_data, 142
 MEASURES, 142
 measures_init, 143
 print_measurement, 143
 measures_init
 measures.c, 140
 measures.h, 143
 netlib.c, 102
 netsim.c
 event_setup, 132
 netsim_start, 132
 pisas_sched, 133
 netsim.h
 event_setup, 134
 netsim_start, 135
 netsim/conf/config.c, 102
 netsim/conf/config.h, 105
 netsim/csma.c, 108
 netsim/csma.h, 118
 netsim/event.c, 119
 netsim/event.h, 125
 netsim/netsim.c, 131
 netsim/netsim.h, 134
 netsim/queues/fifo.c, 136
 netsim/queues/fifo.h, 137
 netsim/queues/measures.c, 139
 netsim/queues/measures.h, 141
 netsim/queues/packet.c, 143
 netsim/queues/packet.h, 149
 netsim/queues/queue_man.c, 155
 netsim/queues/queue_man.h, 158
 netsim/sys_aqueue.c, 161
 netsim/sys_aqueue.h, 168
 netsim_start
 netsim.c, 132
 netsim.h, 135
 network, 31
 network.c, 170
 network.h, 171
 network_node_list, 32
 PACKET
 packet.h, 150
 packet, 32
 packet.c
 measurement_self_collect_data, 144
 packet_init, 144
 packet_list_config, 145
 packet_list_get_first, 145
 packet_list_init, 146
 packet_list_insert_packet, 146
 packet_list_is_empty, 147
 packet_list_new_packet, 147
 packet_list_remove_packet, 148
 test_packet_list_new_packet, 148
 packet.h
 measurement_self_collect_data, 151
 PACKET, 150
 PACKET_INFO, 150
 packet_init, 151
 PACKET_LIST, 150
 packet_list_config, 152
 packet_list_get_first, 152
 packet_list_init, 152
 packet_list_insert_packet, 153
 packet_list_is_empty, 153
 packet_list_new_packet, 154
 packet_list_remove_packet, 154
 test_packet_list_new_packet, 155
 PACKET_INFO
 packet.h, 150
 packet_info, 33

packet_init
 packet.c, 144
 packet.h, 151
PACKET_LIST
 packet.h, 150
packet_list, 34
packet_list_config
 packet.c, 145
 packet.h, 152
packet_list_get_first
 packet.c, 145
 packet.h, 152
packet_list_init
 packet.c, 146
 packet.h, 152
packet_list_insert_packet
 packet.c, 146
 packet.h, 153
packet_list_is_empty
 packet.c, 147
 packet.h, 153
packet_list_new_packet
 packet.c, 147
 packet.h, 154
packet_list_remove_packet
 packet.c, 148
 packet.h, 154
pisas_sched
 netsim.c, 133
print_event_list
 event.c, 124
print_list
 linked_list.c, 93
 linked_list.h, 101
print_measurement
 measures.c, 141
 measures.h, 143
print_statistical_value
 measures.c, 140
pvalue_chi_id
 quantile.h, 172
pvalue_normal_id
 quantile.h, 173
quantile.h, 172
 pvalue_chi_id, 172
 pvalue_normal_id, 173
QUEUE_CONF
 config.h, 106
queue_config, 36
QUEUE_MAN
 queue_man.h, 159
queue_man.c
 queue_man_activate_type, 156
 queue_man_init, 156
 queue_man_register_new_type, 157
 queue_man_unregister_type, 157
queue_man.h
 QUEUE_MAN, 159
 queue_man_activate_type, 159
 queue_man_init, 159
 queue_man_register_new_type, 160
 queue_man_unregister_type, 160
queue_man_activate_type
 queue_man.c, 156
 queue_man.h, 159
queue_man_init
 queue_man.c, 156
 queue_man.h, 159
queue_man_register_new_type
 queue_man.c, 157
 queue_man.h, 160
queue_man_unregister_type
 queue_man.c, 157
 queue_man.h, 160
queue_type, 36
queue_type_list, 38
RANDOM_CONF
 config.h, 106
random_config, 39
random_distribution, 40
rnddist.c
 _composition_gen_rnd, 73
 _convolution_gen_rnd, 73
 _inverse_bernoulli, 74
 _inverse_empirical, 74
 _inverse_exp, 74
 _inverse_gen_rnd, 75
 _inverse_geometric, 75
 _inverse_int_uniform, 75
 _inverse_pareto, 76
 _inverse_uniform, 76
 _inverse_weibull, 76
 _irand_gen_bernoulli, 76
 _irand_gen_erlang, 77
 _irand_gen_exp, 77
 _irand_gen_int_uniform, 78
 _irand_gen_pareto, 78
 _irand_gen_uniform, 79

rndnum.c
 _combined_linear_gen_init, 81
 _combined_linear_gen_num, 81
 _combined_linear_gen_real, 82
 _linear_lehmer_gen_init, 82
 _linear_lehmer_gen_num, 83
 _linear_lehmer_gen_real, 83
 COMBINED_LINEAR_GEN, 81
 irand_gen_random, 83
 irand_gen_random_real, 84
 irand_gen_random, 84
 irand_gen_random_real, 85
 irand_init, 85
 irand_new_seed, 86
 irand_random_seed, 86
 LINEAR_LEHMER_GEN, 81
 row, 41
 runtime_state
 config, 13

 shortest_path_distance, 42
 sp_distance_list, 43
 sparse_matrix, 44
 STAT_NUM
 stat_num.h, 176
 stat_num.c, 173
 stat_num_init, 174
 stat_num_new_sample, 174
 stat_num_new_time_sample, 174
 stat_num.h, 175
 STAT_NUM, 176
 stat_num_init, 176
 stat_num_new_sample, 176
 stat_num_new_time_sample, 176
 stat_num_init
 stat_num.c, 174
 stat_num.h, 176
 stat_num_new_sample
 stat_num.c, 174
 stat_num.h, 176
 stat_num_new_time_sample
 stat_num.c, 174
 stat_num.h, 176
 statistical_number, 44
 STOP_CONF
 config.h, 106
 stop_config, 45
 swap_prev_event
 event.h, 126
 sys_aqueue.c
 _allow_continue, 162
 _generate_arrival, 162
 _generate_end_service, 163
 _generate_event, 163
 _get_next_event, 164
 _packet_from_event, 165
 _process_arrival, 165
 _process_end_service, 166
 _process_event, 166
 _remove_event, 167
 sys_state_init, 167
 sys_aqueue.h
 SYS_STATE, 169
 sys_state_init, 169
 SYS_STATE
 sys_aqueue.h, 169
 sys_state, 46
 sys_state_init
 sys_aqueue.c, 167
 sys_aqueue.h, 169
 system_state_operations, 48

 test_chisqr
 chisqr.c, 177
 test_event_list_insert
 event.c, 124
 event.h, 131
 test_packet_list_new_packet
 packet.c, 148
 packet.h, 155
 test_params, 51
 tests/chisqr.c, 177
 TIME
 def.h, 57
 time, 51
 trash, 52
 trash_clean
 error.c, 59
 error.h, 63
 trash_collect_garbage
 error.c, 59
 error.h, 64
 try
 error.h, 62

 uniform_params, 52

 weibull_params, 53

 yy_bs_column

yy_buffer_state, [54](#)
yy_bs_lineno
 yy_buffer_state, [54](#)
yy_buffer_state, [53](#)
 yy_bs_column, [54](#)
 yy_bs_lineno, [54](#)
yy_trans_info, [54](#)
yyalloc, [55](#)
YYLTYPE, [55](#)
YYSTYPE, [56](#)