

QILFAU
0.0

Generated by Doxygen 1.7.3

Mon May 30 2011 10:26:47

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	combined_linear_congruential_gen Struct Reference	5
3.1.1	Detailed Description	6
3.2	config Struct Reference	6
3.2.1	Detailed Description	7
3.2.2	Field Documentation	7
3.2.2.1	runtime_state	7
3.3	csma_conf Struct Reference	8
3.4	csma_state Struct Reference	9
3.5	edge Struct Reference	10
3.6	empirical_params Struct Reference	10
3.6.1	Detailed Description	11
3.7	event Struct Reference	11
3.7.1	Detailed Description	13
3.8	event_info Struct Reference	13
3.8.1	Detailed Description	15
3.9	event_list Struct Reference	15
3.9.1	Detailed Description	16
3.10	fifo_queue Struct Reference	16
3.10.1	Detailed Description	18
3.11	linear_congruential_gen Struct Reference	18
3.11.1	Detailed Description	19
3.12	linked_list Struct Reference	19
3.12.1	Detailed Description	19
3.13	linked_list_manager Struct Reference	20
3.13.1	Detailed Description	20
3.14	matrix Struct Reference	21
3.15	measures Struct Reference	21
3.15.1	Detailed Description	23
3.16	network Struct Reference	23
3.17	packet Struct Reference	24
3.17.1	Detailed Description	25
3.18	packet_info Struct Reference	25

3.18.1	Detailed Description	26
3.19	packet_list Struct Reference	26
3.19.1	Detailed Description	27
3.20	queue_config Struct Reference	28
3.20.1	Detailed Description	28
3.21	queue_type Struct Reference	28
3.21.1	Detailed Description	30
3.22	queue_type_list Struct Reference	30
3.22.1	Detailed Description	31
3.23	random_config Struct Reference	31
3.23.1	Detailed Description	32
3.24	random_distribution Struct Reference	32
3.24.1	Detailed Description	33
3.25	row Struct Reference	33
3.26	shortest_path_dijkstra Struct Reference	34
3.27	statistical_number Struct Reference	34
3.27.1	Detailed Description	35
3.28	stop_config Struct Reference	35
3.28.1	Detailed Description	36
3.29	sys_state Struct Reference	36
3.29.1	Detailed Description	38
3.30	system_state_operations Struct Reference	38
3.30.1	Detailed Description	40
3.31	test_params Struct Reference	41
3.32	time Union Reference	41
3.32.1	Detailed Description	42
3.33	uniform_params Struct Reference	42
3.33.1	Detailed Description	42
3.34	weibull_params Struct Reference	42
3.34.1	Detailed Description	43
3.35	yy_buffer_state Struct Reference	43
3.35.1	Field Documentation	43
3.35.1.1	yy_bs_column	43
3.35.1.2	yy_bs_lineno	43
3.36	yy_trans_info Struct Reference	43
3.37	yyalloc Union Reference	44
3.38	YYLTYPE Struct Reference	44
3.39	YYSTYPE Union Reference	45
4	File Documentation	47
4.1	def.h File Reference	47
4.1.1	Detailed Description	47
4.1.2	Typedef Documentation	47
4.1.2.1	TIME	47
4.2	error.h File Reference	47
4.2.1	Detailed Description	49
4.2.2	Define Documentation	49
4.2.2.1	check_null_pointer	49
4.2.2.2	iprintf	49
4.2.2.3	try	49

4.3	irand/irand.h File Reference	50
4.3.1	Detailed Description	51
4.3.2	Function Documentation	51
4.3.2.1	irand_gen_bernoulli	51
4.3.2.2	irand_gen_erlang	52
4.3.2.3	irand_gen_exp	52
4.3.2.4	irand_gen_int_uniform	52
4.3.2.5	irand_gen_pareto	53
4.3.2.6	irand_gen_random	53
4.3.2.7	irand_gen_random_real	54
4.3.2.8	irand_gen_srandom	54
4.3.2.9	irand_gen_srandom_real	55
4.3.2.10	irand_gen_uniform	55
4.3.2.11	irand_init	56
4.3.2.12	irand_new_seed	56
4.3.2.13	irand_random_seed	56
4.4	irand/rnddist.c File Reference	57
4.4.1	Detailed Description	58
4.4.2	Function Documentation	58
4.4.2.1	_composition_gen_rnd	58
4.4.2.2	_convolution_gen_rnd	58
4.4.2.3	_inverse_bernoulli	59
4.4.2.4	_inverse_empirical	59
4.4.2.5	_inverse_exp	60
4.4.2.6	_inverse_gen_rnd	60
4.4.2.7	_inverse_geometric	60
4.4.2.8	_inverse_int_uniform	61
4.4.2.9	_inverse_pareto	61
4.4.2.10	_inverse_uniform	61
4.4.2.11	_inverse_weibull	61
4.4.2.12	irand_gen_bernoulli	62
4.4.2.13	irand_gen_erlang	62
4.4.2.14	irand_gen_exp	63
4.4.2.15	irand_gen_int_uniform	63
4.4.2.16	irand_gen_pareto	63
4.4.2.17	irand_gen_uniform	64
4.5	irand/rndnum.c File Reference	64
4.5.1	Detailed Description	65
4.5.2	Typedef Documentation	66
4.5.2.1	COMBINED_LINEAR_GEN	66
4.5.2.2	LINEAR_LEHMER_GEN	66
4.5.3	Function Documentation	66
4.5.3.1	_combined_linear_gen_init	66
4.5.3.2	_combined_linear_gen_num	66
4.5.3.3	_combined_linear_gen_real	67
4.5.3.4	_linear_lehmer_gen_init	67
4.5.3.5	_linear_lehmer_gen_num	68
4.5.3.6	_linear_lehmer_gen_real	68
4.5.3.7	irand_gen_random	68
4.5.3.8	irand_gen_random_real	69

4.5.3.9	irand_gen_random	69
4.5.3.10	irand_gen_random_real	70
4.5.3.11	irand_init	70
4.5.3.12	irand_new_seed	71
4.5.3.13	irand_random_seed	71
4.6	list/linked_list.c File Reference	72
4.6.1	Detailed Description	72
4.7	list/linked_list.h File Reference	72
4.7.1	Detailed Description	74
4.7.2	Define Documentation	74
4.7.2.1	container_of	74
4.7.3	Typedef Documentation	74
4.7.3.1	LINKED_LIST	74
4.7.3.2	LINKED_LIST_MAN	74
4.8	matrix/matrix.c File Reference	74
4.8.1	Detailed Description	75
4.9	matrix/matrix.h File Reference	75
4.9.1	Detailed Description	76
4.10	netlib.c File Reference	76
4.10.1	Detailed Description	76
4.11	netsim/conf/config.c File Reference	76
4.11.1	Detailed Description	77
4.11.2	Function Documentation	77
4.11.2.1	config_init	77
4.11.2.2	config_parse_file	77
4.11.2.3	config_random_conf	78
4.11.2.4	config_setup	78
4.12	netsim/conf/config.h File Reference	79
4.12.1	Detailed Description	80
4.12.2	Typedef Documentation	80
4.12.2.1	CONFIG	80
4.12.2.2	QUEUE_CONF	80
4.12.2.3	RANDOM_CONF	80
4.12.2.4	STOP_CONF	80
4.12.3	Function Documentation	80
4.12.3.1	config_init	80
4.12.3.2	config_parse_file	81
4.12.3.3	config_random_conf	81
4.12.3.4	config_setup	81
4.13	netsim/csma.c File Reference	82
4.13.1	Detailed Description	83
4.13.2	Function Documentation	83
4.13.2.1	csma_allow_continue	83
4.13.2.2	csma_generate_access	84
4.13.2.3	csma_generate_arrival	84
4.13.2.4	csma_generate_collision	85
4.13.2.5	csma_generate_end_service	86
4.13.2.6	csma_generate_event	86
4.13.2.7	csma_get_next_event	87
4.13.2.8	csma_process_access_event	87

4.13.2.9	csma_process_arrival	88
4.13.2.10	csma_process_collision	89
4.13.2.11	csma_process_end_service	89
4.13.2.12	csma_process_event	90
4.13.2.13	csma_remove_event	91
4.13.2.14	csma_state_init	91
4.14	netsim/csma.h File Reference	92
4.14.1	Detailed Description	93
4.14.2	Function Documentation	93
4.14.2.1	csma_state_init	93
4.15	netsim/event.c File Reference	94
4.15.1	Detailed Description	94
4.15.2	Function Documentation	95
4.15.2.1	event_init	95
4.15.2.2	event_list_get_first	95
4.15.2.3	event_list_init	95
4.15.2.4	event_list_insert_event	95
4.15.2.5	event_list_is_empty	96
4.15.2.6	event_list_new_event	96
4.15.2.7	event_list_remove_event	97
4.15.2.8	event_save	97
4.15.2.9	print_event_list	97
4.15.2.10	test_event_list_insert	97
4.16	netsim/event.h File Reference	98
4.16.1	Detailed Description	99
4.16.2	Define Documentation	100
4.16.2.1	swap_prev_event	100
4.16.3	Typedef Documentation	100
4.16.3.1	EVENT	100
4.16.3.2	EVENT_LIST	100
4.16.3.3	EVENTINFO	100
4.16.4	Function Documentation	100
4.16.4.1	event_init	100
4.16.4.2	event_list_get_first	101
4.16.4.3	event_list_init	101
4.16.4.4	event_list_insert_event	101
4.16.4.5	event_list_is_empty	101
4.16.4.6	event_list_new_event	102
4.16.4.7	event_list_remove_event	102
4.16.4.8	event_save	102
4.16.4.9	test_event_list_insert	103
4.17	netsim/netsim.c File Reference	103
4.17.1	Detailed Description	104
4.17.2	Function Documentation	104
4.17.2.1	event_setup	104
4.17.2.2	netsim_start	105
4.17.2.3	pisas_sched	105
4.18	netsim/netsim.h File Reference	106
4.18.1	Detailed Description	106
4.18.2	Function Documentation	106

4.18.2.1	event_setup	106
4.18.2.2	netsim_start	107
4.19	netsim/sys_aqueue.c File Reference	108
4.19.1	Detailed Description	108
4.19.2	Function Documentation	108
4.19.2.1	_allow_continue	108
4.19.2.2	_generate_arrival	109
4.19.2.3	_generate_end_service	110
4.19.2.4	_generate_event	110
4.19.2.5	_get_next_event	111
4.19.2.6	_packet_from_event	111
4.19.2.7	_process_arrival	112
4.19.2.8	_process_end_service	112
4.19.2.9	_process_event	113
4.19.2.10	_remove_event	113
4.19.2.11	sys_state_init	114
4.20	netsim/sys_aqueue.h File Reference	115
4.20.1	Detailed Description	116
4.20.2	Typedef Documentation	116
4.20.2.1	SYS_STATE	116
4.20.3	Function Documentation	116
4.20.3.1	sys_state_init	116
4.21	network.c File Reference	117
4.21.1	Detailed Description	117
4.22	network.h File Reference	118
4.22.1	Detailed Description	118
4.23	quantile.h File Reference	118
4.23.1	Detailed Description	119
4.23.2	Define Documentation	119
4.23.2.1	pvalue_chi_id	119
4.23.2.2	pvalue_normal_id	119
4.24	queues/fifo.c File Reference	119
4.24.1	Detailed Description	120
4.24.2	Function Documentation	120
4.24.2.1	fifo_init	120
4.24.2.2	fifo_setup	121
4.25	queues/fifo.h File Reference	121
4.25.1	Detailed Description	122
4.25.2	Typedef Documentation	122
4.25.2.1	FIFO_QINFO	122
4.25.3	Function Documentation	122
4.25.3.1	fifo_init	122
4.25.3.2	fifo_setup	123
4.26	queues/measures.c File Reference	123
4.26.1	Detailed Description	123
4.26.2	Define Documentation	124
4.26.2.1	print_statistical_value	124
4.26.3	Function Documentation	124
4.26.3.1	measurement_collect_data	124
4.26.3.2	measures_init	125

4.26.3.3	print_measurement	125
4.27	queues/measures.h File Reference	125
4.27.1	Detailed Description	125
4.27.2	Typedef Documentation	126
4.27.2.1	MEASURES	126
4.27.3	Function Documentation	126
4.27.3.1	measurement_collect_data	126
4.27.3.2	measures_init	127
4.27.3.3	print_measurement	127
4.28	queues/packet.c File Reference	127
4.28.1	Detailed Description	128
4.28.2	Function Documentation	128
4.28.2.1	measurement_self_collect_data	128
4.28.2.2	packet_init	128
4.28.2.3	packet_list_config	129
4.28.2.4	packet_list_get_first	129
4.28.2.5	packet_list_init	129
4.28.2.6	packet_list_insert_packet	130
4.28.2.7	packet_list_is_empty	130
4.28.2.8	packet_list_new_packet	130
4.28.2.9	packet_list_remove_packet	131
4.28.2.10	test_packet_list_new_packet	131
4.29	queues/packet.h File Reference	132
4.29.1	Detailed Description	133
4.29.2	Typedef Documentation	133
4.29.2.1	PACKET	133
4.29.2.2	PACKET_INFO	133
4.29.2.3	PACKET_LIST	133
4.29.3	Function Documentation	133
4.29.3.1	measurement_self_collect_data	133
4.29.3.2	packet_init	134
4.29.3.3	packet_list_config	134
4.29.3.4	packet_list_get_first	134
4.29.3.5	packet_list_init	135
4.29.3.6	packet_list_insert_packet	135
4.29.3.7	packet_list_is_empty	135
4.29.3.8	packet_list_new_packet	136
4.29.3.9	packet_list_remove_packet	136
4.29.3.10	test_packet_list_new_packet	137
4.30	queues/queue_man.c File Reference	137
4.30.1	Detailed Description	137
4.30.2	Function Documentation	138
4.30.2.1	queue_man_activate_type	138
4.30.2.2	queue_man_init	138
4.30.2.3	queue_man_register_new_type	138
4.30.2.4	queue_man_unregister_type	138
4.31	queues/queue_man.h File Reference	139
4.31.1	Detailed Description	139
4.31.2	Typedef Documentation	140
4.31.2.1	QUEUE_MAN	140

4.31.3	Function Documentation	140
4.31.3.1	queue_man_activate_type	140
4.31.3.2	queue_man_init	140
4.31.3.3	queue_man_register_new_type	140
4.31.3.4	queue_man_unregister_type	141
4.32	random.h File Reference	141
4.32.1	Detailed Description	142
4.32.2	Function Documentation	142
4.32.2.1	gen_bernoulli	142
4.32.2.2	gen_exponential	142
4.32.2.3	gen_int_uniform	143
4.32.2.4	gen_normal	143
4.32.2.5	gen_poisson	144
4.32.2.6	gen_uniform	144
4.32.2.7	random_dist_init_bernoulli0	144
4.32.2.8	random_dist_init_bernoulli1	144
4.32.2.9	random_dist_init_exp0	145
4.32.2.10	random_dist_init_exp1	145
4.32.2.11	random_dist_init_file0	145
4.32.2.12	random_dist_init_uniform0	146
4.32.2.13	random_dist_init_uniform1	146
4.32.2.14	random_init	146
4.33	stat_num.c File Reference	147
4.33.1	Detailed Description	148
4.33.2	Function Documentation	148
4.33.2.1	calc_avg	148
4.33.2.2	stat_num_init	148
4.33.2.3	stat_num_new_sample	148
4.33.2.4	stat_num_new_time_sample	149
4.34	stat_num.h File Reference	149
4.34.1	Detailed Description	149
4.34.2	Typedef Documentation	150
4.34.2.1	STAT_NUM	150
4.34.3	Function Documentation	150
4.34.3.1	stat_num_init	150
4.34.3.2	stat_num_new_sample	150
4.34.3.3	stat_num_new_time_sample	150
4.35	tests/chisqr.c File Reference	151
4.35.1	Detailed Description	151
4.35.2	Function Documentation	151
4.35.2.1	test_chisqr	151

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

combined_linear_congruential_gen	5
config	6
csma_conf	8
csma_state	9
edge	10
empirical_params (Parameters of Empirical Discrete distribution)	10
event	11
event_info	13
event_list	15
fifo_queue	16
linear_congruential_gen	18
linked_list	19
linked_list_manager	20
matrix	21
measures	21
network	23
packet	24
packet_info	25
packet_list	26
queue_config	28
queue_type	28
queue_type_list	30
random_config	31
random_distribution (Random distribution structure (a framework))	32
row	33
shortest_path_dijkstra	34
statistical_number	34
stop_config	35
sys_state	36

system_state_operations (Functions of a simulated system)	38
test_params	41
time	41
uniform_params (Parameters of Uniform distribution)	42
weibull_params (Parameters of Weibull distribution)	42
yy_buffer_state	43
yy_trans_info	43
yyalloc	44
YYLTYPE	44
YYSTYPE	45

Chapter 2

File Index

2.1 File List

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

def.h	47
error.h	47
netlib.c	76
network.c	117
network.h	118
quantile.h	118
random.h	141
stat_num.c	147
stat_num.h	149
irand/irand.h	50
irand/rnndist.c	57
irand/rndnum.c	64
list/linked_list.c	72
list/linked_list.h	72
matrix/matrix.c	74
matrix/matrix.h	75
netsim/csma.c	82
netsim/csma.h	92
netsim/event.c	94
netsim/event.h	98
netsim/netsim.c	103
netsim/netsim.h	106
netsim/sys_aqueue.c	108
netsim/sys_aqueue.h	115
netsim/conf/config.c	76
netsim/conf/config.h	79
netsim/conf/lexer.h	??
netsim/conf/parser.h	??
queues/fifo.c	119

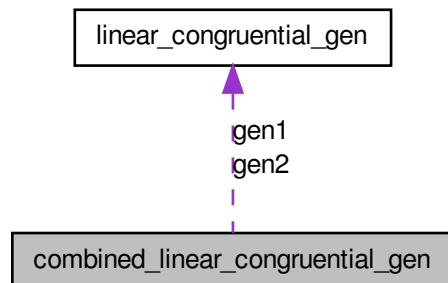
queues/ fifo.h	121
queues/ measures.c	123
queues/ measures.h	125
queues/ packet.c	127
queues/ packet.h	132
queues/ queue_man.c	137
queues/ queue_man.h	139
ranlib/ ranlib.h	??
rng/ rngs.h	??
rng/ rvgs.h	??
tests/ chisqr.c	151

Chapter 3

Data Structure Documentation

3.1 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`

3.1.1 Detailed Description

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

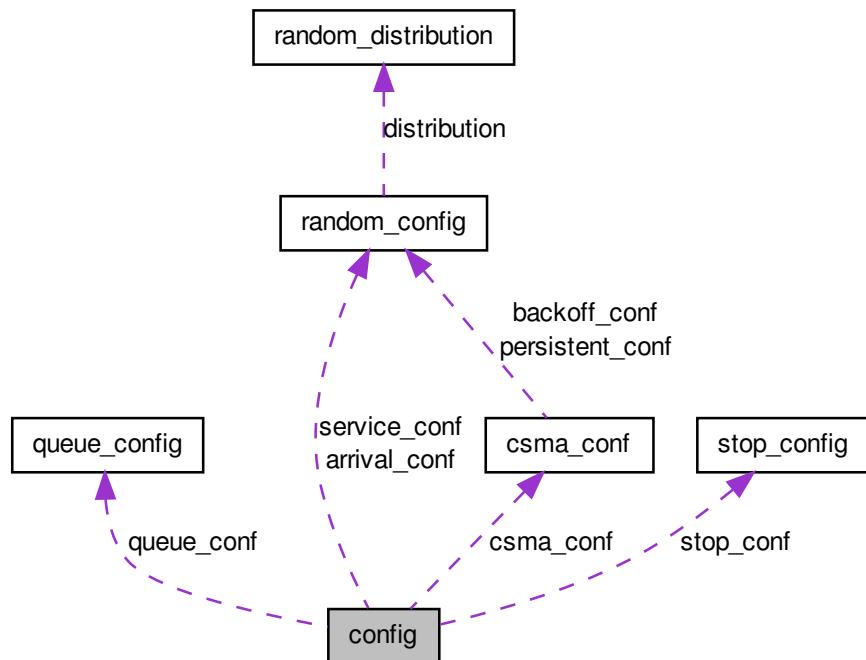
The documentation for this struct was generated from the following file:

- irand/rndnum.c

3.2 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`

3.2.1 Detailed Description

Configuration from user

3.2.2 Field Documentation

3.2.2.1 void* config::runtime_state

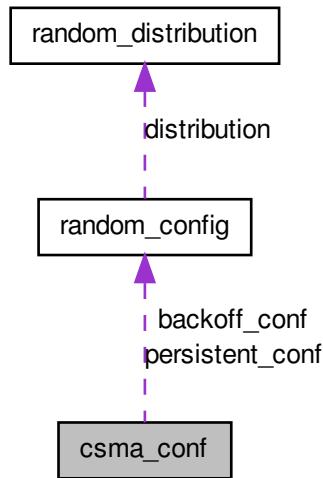
Current simulation: used for handling signal SIGINT (we dont want to wait too long time, but also want to see the intermediate result)

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

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

3.3 csma_conf Struct Reference

Collaboration diagram for csma_conf:



Data Fields

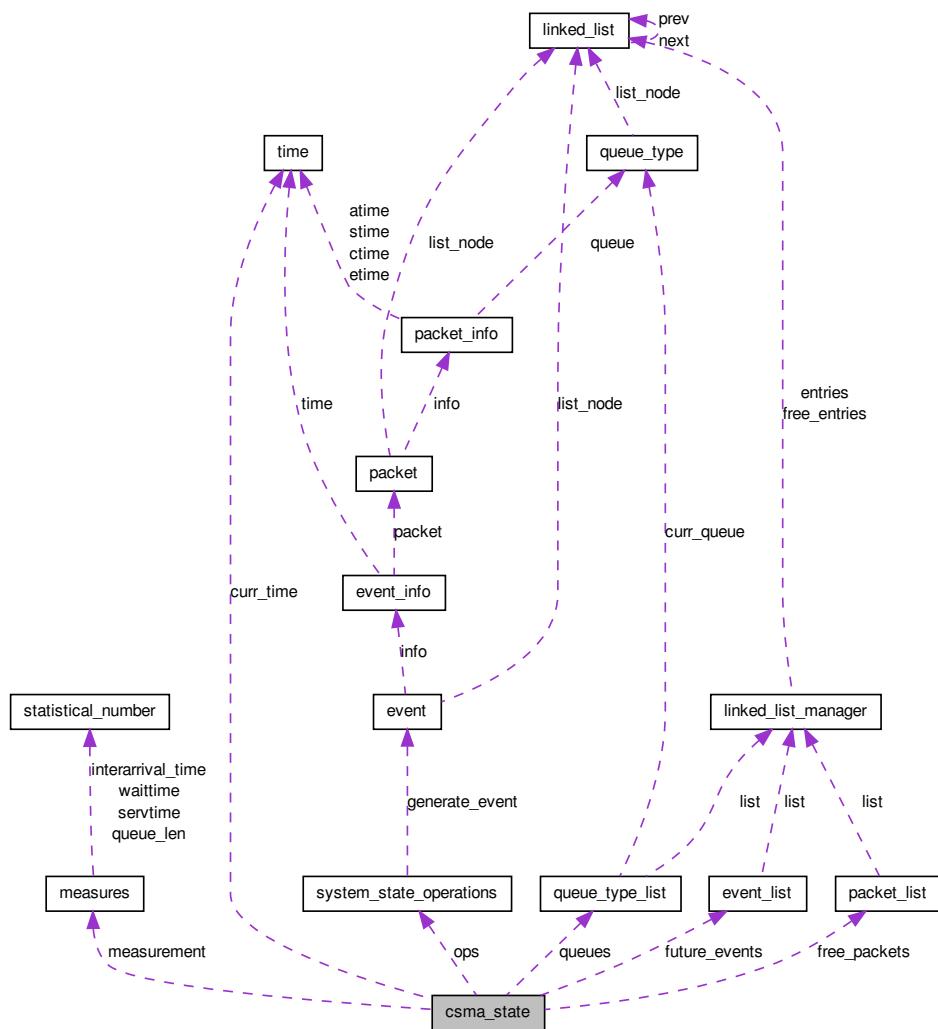
- double `slot_time`
slot time
- double `collision_time`
Collision time.
- int `nstations`
number of queues or station
- `RANDOM_CONF` `backoff_conf`
Backoff.
- `RANDOM_CONF` `persistent_conf`
persistent probability

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

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

3.4 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

3.5 edge Struct Reference

Data Fields

- int **source**
- int **dest**
- void * **data**

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

- matrix/matrix.h

3.6 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.

3.6.1 Detailed Description

Parameters of Empirical Discrete distribution.

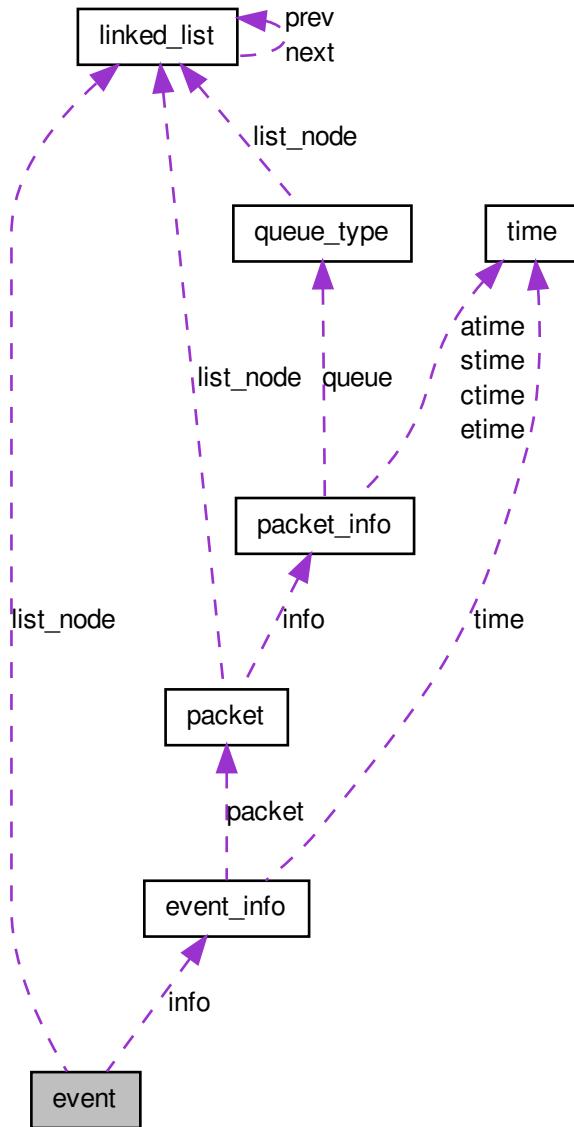
The documentation for this struct was generated from the following file:

- irand/[irand.h](#)

3.7 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.

3.7.1 Detailed Description

Event structure

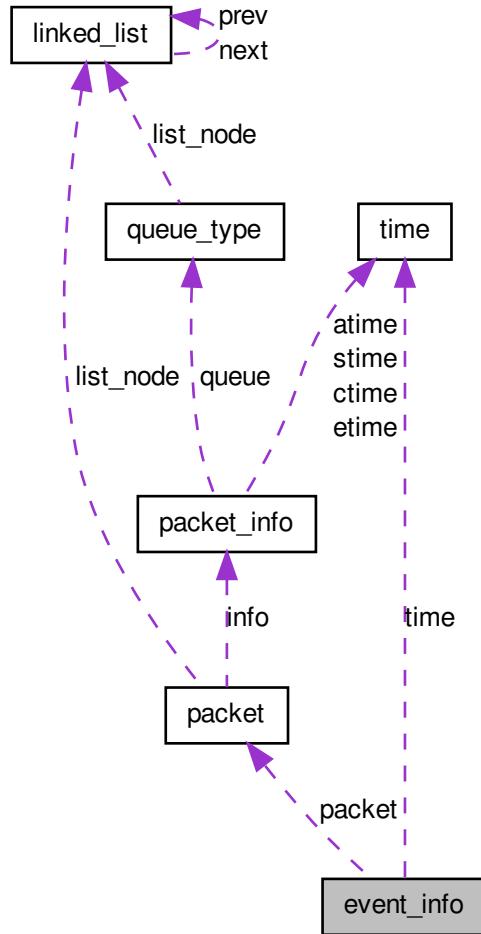
The documentation for this struct was generated from the following file:

- netsim/[event.h](#)

3.8 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)

3.8.1 Detailed Description

Information in an event

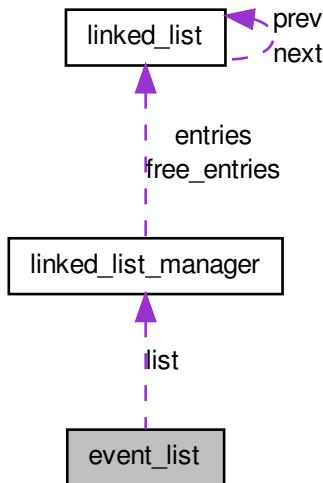
The documentation for this struct was generated from the following file:

- netsim/[event.h](#)

3.9 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 *,...)`

3.9.1 Detailed Description

Event list structure

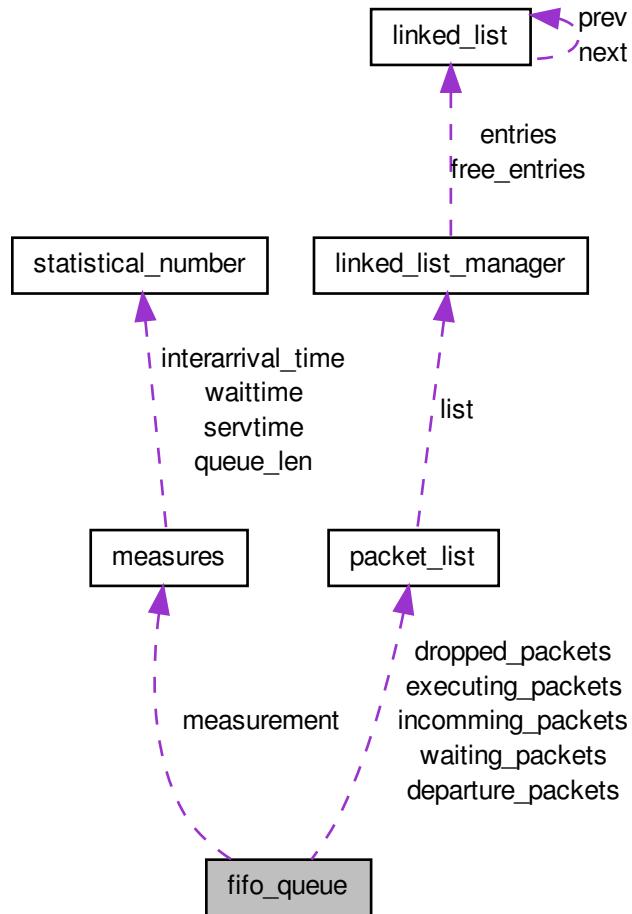
The documentation for this struct was generated from the following file:

- netsim/[event.h](#)

3.10 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.

3.10.1 Detailed Description

FIFO queue structure

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

- queues/[fifo.h](#)

3.11 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.

3.11.1 Detailed Description

Linear Congruential Generator

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

- irand/[rndnum.c](#)

3.12 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.

3.12.1 Detailed Description

Linked list structure

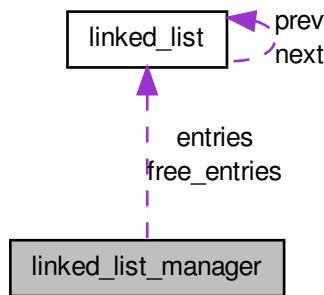
The documentation for this struct was generated from the following file:

- list/[linked_list.h](#)

3.13 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.

3.13.1 Detailed Description

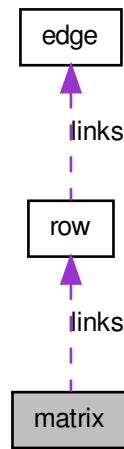
Linked list manager. Manage not only nodes of list but also free nodes (used for allocating new node)

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

- list/[linked_list.h](#)

3.14 matrix Struct Reference

Collaboration diagram for matrix:



Data Fields

- int **nnodes**
- **ROW** * **links**

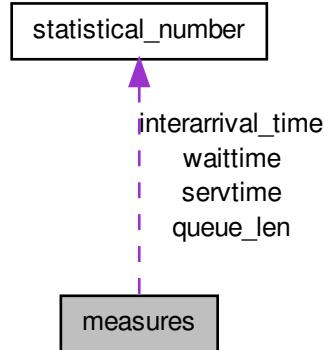
The documentation for this struct was generated from the following file:

- matrix/**matrix.h**

3.15 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).

3.15.1 Detailed Description

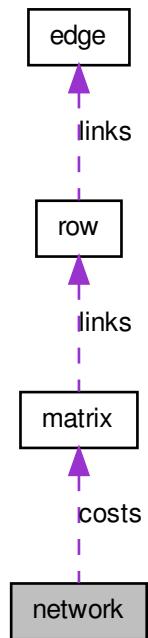
MEASURE structure used to store some results when simulating queue system

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

- [queues/measures.h](#)

3.16 network Struct Reference

Collaboration diagram for network:



Data Fields

- [MATRIX costs](#)

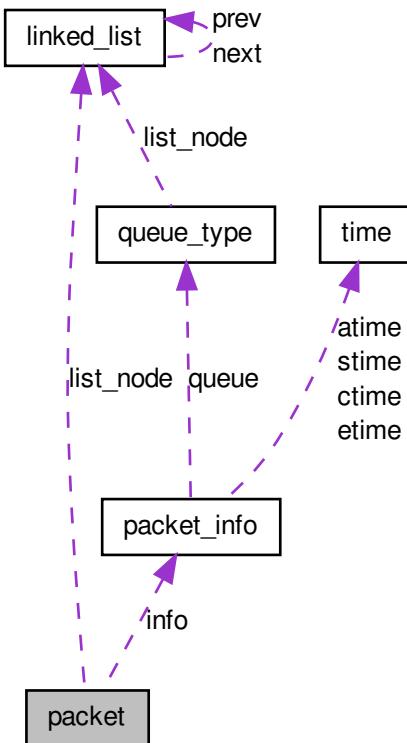
The documentation for this struct was generated from the following file:

- [network.h](#)

3.17 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.

3.17.1 Detailed Description

Packet structure

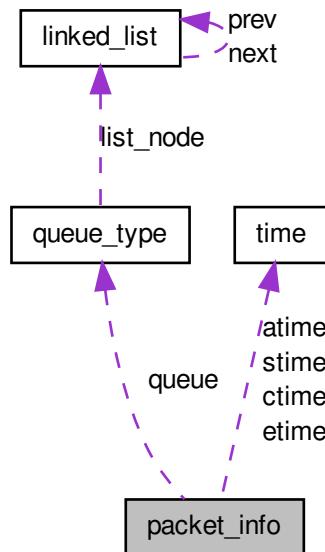
The documentation for this struct was generated from the following file:

- queues/[packet.h](#)

3.18 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)

3.18.1 Detailed Description

Packet information

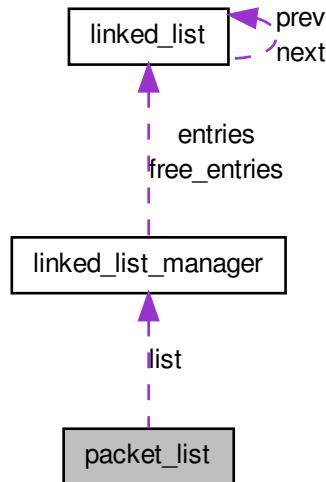
The documentation for this struct was generated from the following file:

- `queues/packet.h`

3.19 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)

3.19.1 Detailed Description

Packet list structure

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

- queues/[packet.h](#)

3.20 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

3.20.1 Detailed Description

Queue configuration

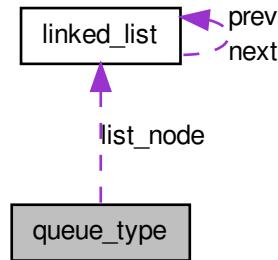
The documentation for this struct was generated from the following file:

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

3.21 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](#)

3.21.1 Detailed Description

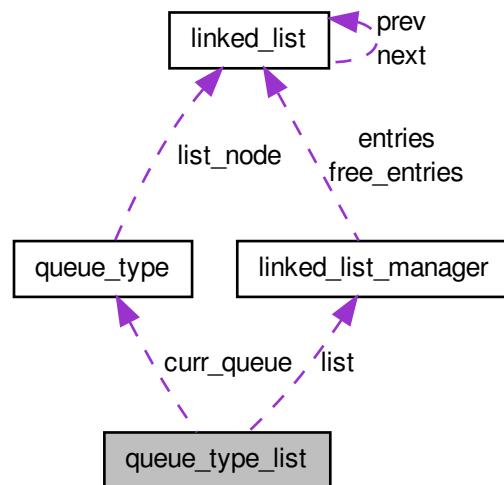
Structure of a queue type. Can be seen as an interface of a queue object
The documentation for this struct was generated from the following file:

- queues/[queue_man.h](#)

3.22 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.

3.22.1 Detailed Description

Queue manager structure

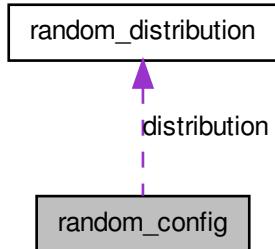
The documentation for this struct was generated from the following file:

- `queues/queue_man.h`

3.23 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.

3.23.1 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](#)

3.24 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.

3.24.1 Detailed Description

Random distribution structure (a framework)

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

- irand/[irand.h](#)

3.25 row Struct Reference

Collaboration diagram for row:



Data Fields

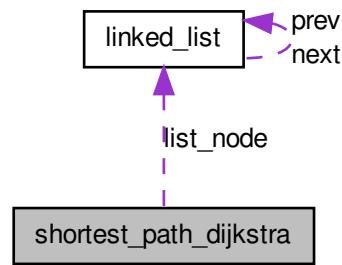
- int **nlinks**
- [EDGE](#) * **links**

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

- matrix/[matrix.h](#)

3.26 shortest_path_dijkstra Struct Reference

Collaboration diagram for shortest_path_dijkstra:



Data Fields

- **LINKED_LIST** **list_node**
- int **node**
- int **last_node**
- double **total_cost**

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

- [network.h](#)

3.27 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.

3.27.1 Detailed Description

Statistical information of a random process

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

- `stat_num.h`

3.28 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.

3.28.1 Detailed Description

Define conditions of stopping program

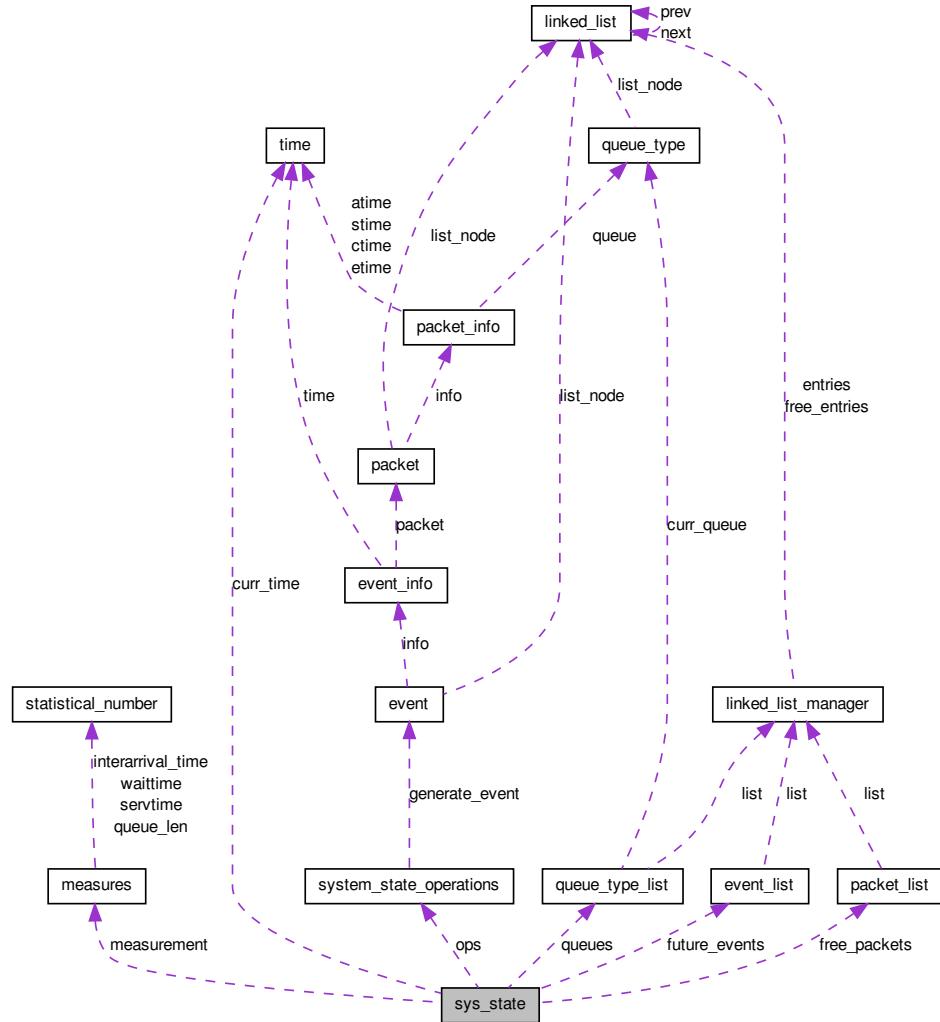
The documentation for this struct was generated from the following file:

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

3.29 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).

3.29.1 Detailed Description

Structure representing the system state in simulation.

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

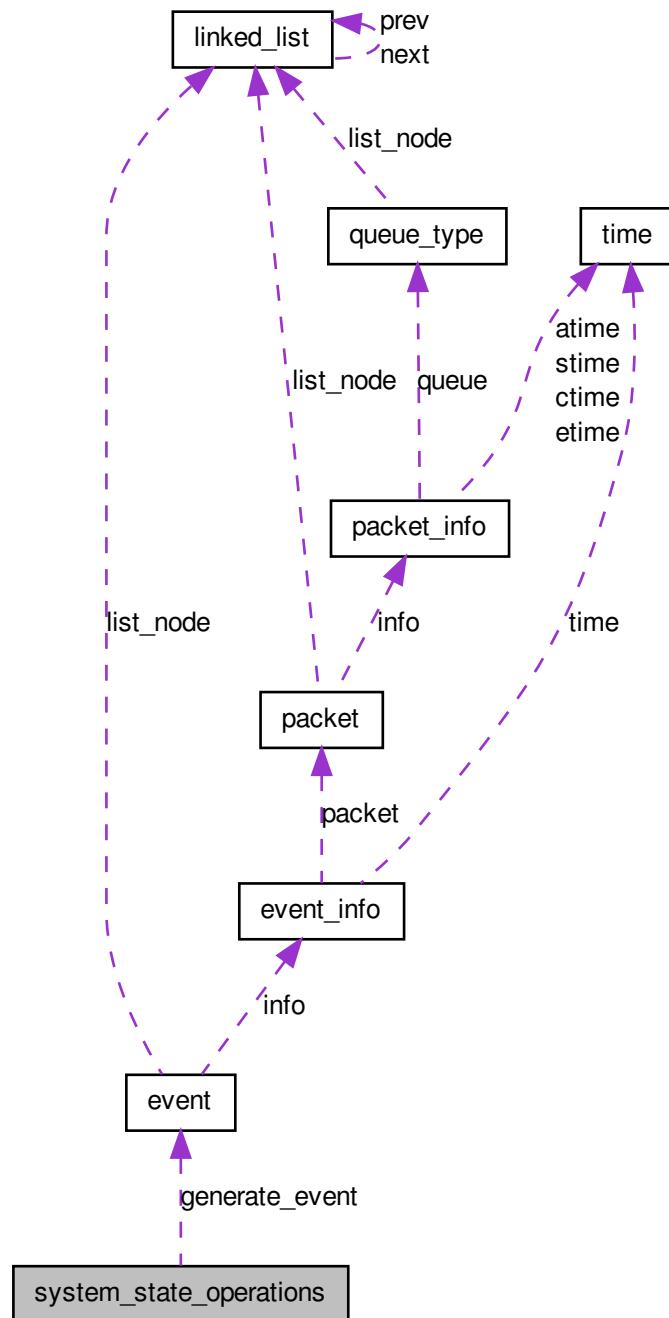
- netsim/[sys_aqueue.h](#)

3.30 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)

3.30.1 Detailed Description

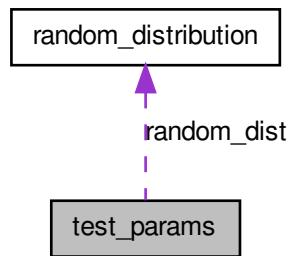
Functions of a simulated system.

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

- netsim/netsim.h

3.31 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](#)

3.32 time Union Reference

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

Data Fields

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

3.32.1 Detailed Description

Time definition

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

- [def.h](#)

3.33 uniform_params Struct Reference

Parameters of Uniform distribution.

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

Data Fields

- double **from**
Lower-bound value.
- double **to**
Upper-bound value.

3.33.1 Detailed Description

Parameters of Uniform distribution.

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

- [irand/irand.h](#)

3.34 weibull_params Struct Reference

Parameters of Weibull distribution.

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

Data Fields

- double **a**
Value of a.
- double **b**
Value of b.

3.34.1 Detailed Description

Parameters of Weibull distribution.

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

- irand/[irand.h](#)

3.35 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**

3.35.1 Field Documentation

3.35.1.1 int yy_buffer_state::yy_bs_column

The column count.

3.35.1.2 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

3.36 yy_trans_info Struct Reference

Data Fields

- flex_int32_t **yy_verify**

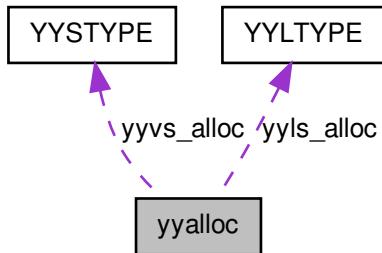
- `flex_int32_t yy_nxt`

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

- `netsim/conf/lexer.c`

3.37 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`

3.38 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

3.39 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`

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.h File Reference

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 **iprintf**(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.

Variables

- long **debug**

4.2.1 Detailed Description

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

Date

Created on: Apr 6, 2011

Author

iizke

4.2.2 Define Documentation**4.2.2.1 #define check_null_pointer(_p)****Value:**

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

4.2.2.2 #define iprintf(level, fmt, args...)**Value:**

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

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

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

4.3 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)

4.3.1 Detailed Description

My implementation of Random number and distribution (called irand)

Date

Created on: Apr 28, 2011

Author

iizke

4.3.2 Function Documentation

4.3.2.1 double `irand_gen_bernoulli (double prob)`

Bernoulli generator of random number

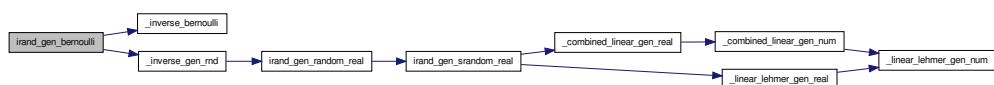
Parameters

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

Returns

A random value (0 or 1)

Here is the call graph for this function:



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

Erlang Generator of random number

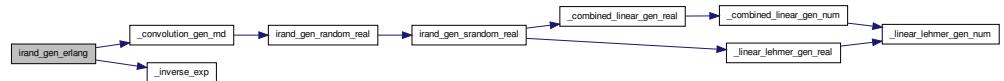
Parameters

<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:



4.3.2.3 double irand_gen_exp (double *lambda*)

Exponential generator of random number

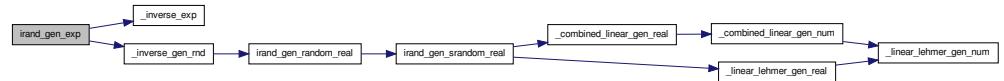
Parameters

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

Returns

A real random value

Here is the call graph for this function:



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

Integer Uniform generator of random number

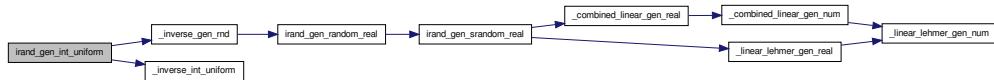
Parameters

<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:

**4.3.2.5 double irand_gen_pareto (double *lambda*)**

Pareto generator of random number

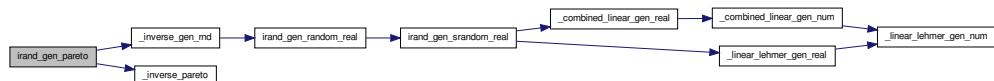
Parameters

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

Returns

A random number

Here is the call graph for this function:

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

Pseudo Integer Random Uniform generator

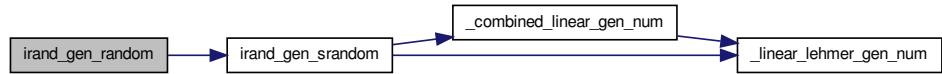
Parameters

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

Returns

Random value

Here is the call graph for this function:

**4.3.2.7 double irand_gen_random_real (int type)**

Pseudo Real-value Random Uniform generator

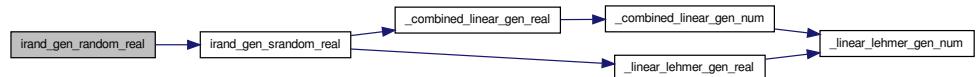
Parameters

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

Returns

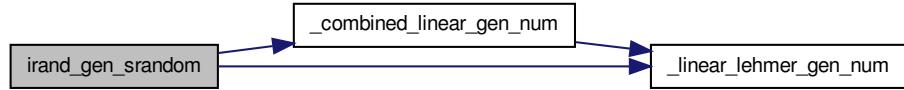
Random value

Here is the call graph for this function:

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

Generate random number in Combined Linear Congruential

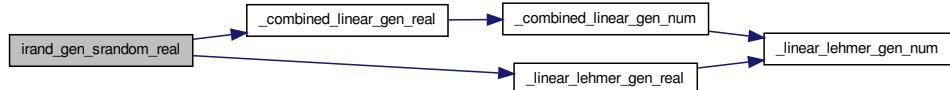
Here is the call graph for this function:



4.3.2.9 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:



4.3.2.10 double irand_gen_uniform (double from, double to)

Uniform generator of random number (real value)

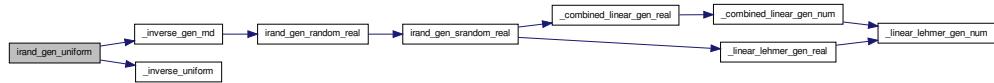
Parameters

<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:



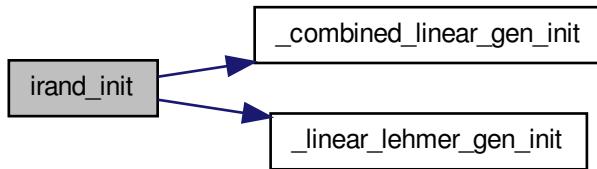
4.3.2.11 int irand_init()

Initialize pseudo random generator

Returns

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

Here is the call graph for this function:

**4.3.2.12 int irand_new_seed(unsigned long seed)**

Pseudo random generator with seed

Parameters

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

Returns

integer pseudo random value

4.3.2.13 int irand_random_seed()

Pseudo random generator with random-selected seed

Returns

integer pseudo random value

Here is the call graph for this function:



4.4 irand/rnndist.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](#) ()

4.4.1 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

4.4.2 Function Documentation

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

Framework of generating random variable based on composition method.

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

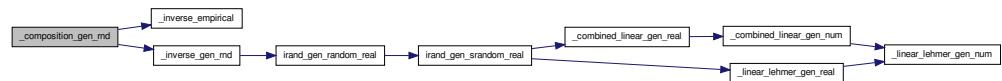
Parameters

<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:



4.4.2.2 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.

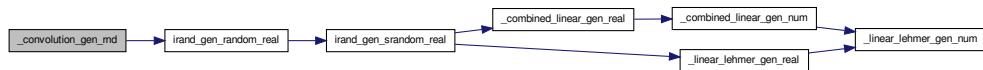
Parameters

<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:



4.4.2.3 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

4.4.2.4 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

4.4.2.5 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

4.4.2.6 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

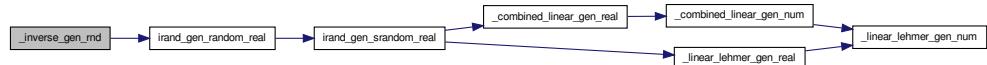
<i>inverse_func</i>	: inverse function (of a particular distribution).
<i>params</i>	: parameters used for function <i>inverse_func</i> .

Returns

random value.

1st step: Generate random integer number

Here is the call graph for this function:



4.4.2.7 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

4.4.2.8 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

4.4.2.9 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

4.4.2.10 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

4.4.2.11 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

4.4.2.12 double irand_gen_bernoulli (double prob)

Bernoulli generator of random number

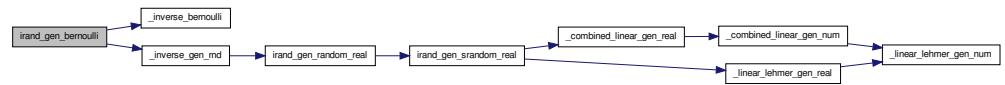
Parameters

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

Returns

A random value (0 or 1)

Here is the call graph for this function:

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

Erlang Generator of random number

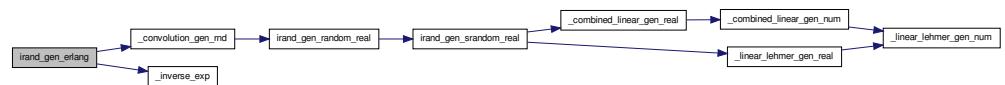
Parameters

<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:



4.4.2.14 double irand_gen_exp (double *lambda*)

Exponential generator of random number

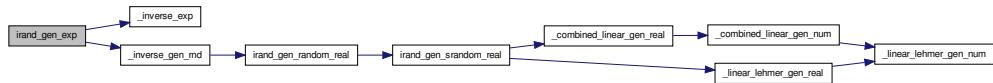
Parameters

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

Returns

A real random value

Here is the call graph for this function:



4.4.2.15 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:



4.4.2.16 double irand_gen_pareto (double *lambda*)

Pareto generator of random number

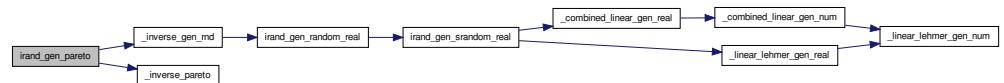
Parameters

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

Returns

A random number

Here is the call graph for this function:

**4.4.2.17 double irand_gen_uniform (double from, double to)**

Uniform generator of random number (real value)

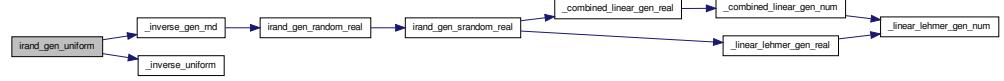
Parameters

<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:

**4.5 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.

4.5.1 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

4.5.2 Typedef Documentation**4.5.2.1 `typedef struct combined_linear_congruential_gen COMBINED_LINEAR_GEN`**

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

4.5.2.2 `typedef struct linear_congruential_gen LINEAR_LEHMER_GEN`

Linear Congruential Generator

4.5.3 Function Documentation**4.5.3.1 `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)

4.5.3.2 `unsigned long _combined_linear_gen_num(COMBINED_LINEAR_GEN * gen)`

Generate a pseudo random number based on Combined Linear Congruential Generator

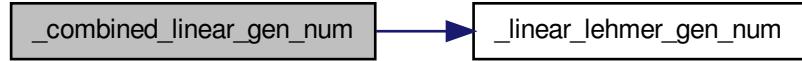
Parameters

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

Returns

Random number

Here is the call graph for this function:



4.5.3.3 double _combined_linear_gen_real (COMBINED_LINEAR_GEN * gen)

generate real number in range [0,1]

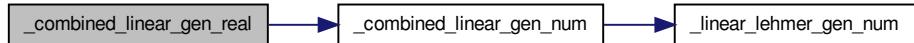
Parameters

gen	: Combined Linear Generator
-----	-----------------------------

Returns

real number

Here is the call graph for this function:



4.5.3.4 int _linear_lehmer_gen_init (LINEAR_LEHMER_GEN * gen)

Initialize the Linear Congruential Generator

Parameters

gen	: Linear Congruential Generator
-----	---------------------------------

Returns

Error code (defined in libs/error.h)

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

Multiplier = $7^5 = 16807$

Increment = 0

4.5.3.5 `unsigned long _linear_lehmer_gen_num (LINEAR_LEHMER_GEN * gen)`

Linear Congruential Generator

Parameters

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

Returns

Random number

4.5.3.6 `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:



4.5.3.7 `unsigned long irand_gen_random (int type)`

Pseudo Integer Random Uniform generator

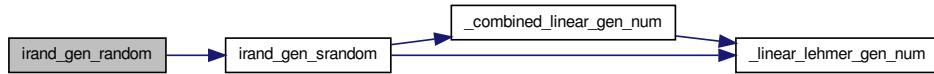
Parameters

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

Returns

Random value

Here is the call graph for this function:

**4.5.3.8 double irand_gen_random_real (int type)**

Pseudo Real-value Random Uniform generator

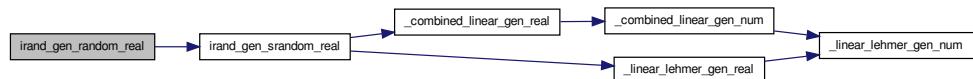
Parameters

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

Returns

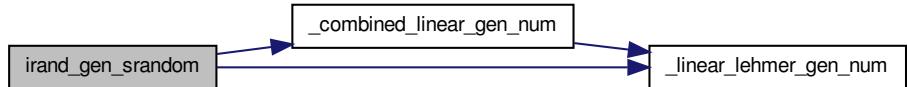
Random value

Here is the call graph for this function:

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

Generate random number in Combined Linear Congruential

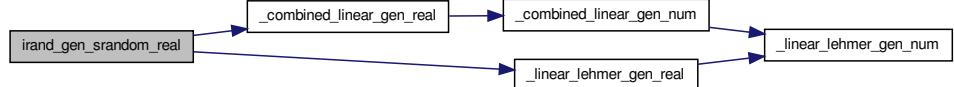
Here is the call graph for this function:



4.5.3.10 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:



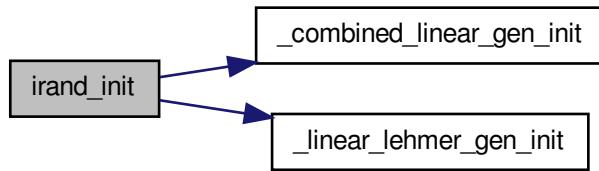
4.5.3.11 int irand_init ()

Initialize pseudo random generator

Returns

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

Here is the call graph for this function:



4.5.3.12 int irand_new_seed(unsigned long seed)

Pseudo random generator with seed

Parameters

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

Returns

integer pseudo random value

4.5.3.13 int irand_random_seed()

Pseudo random generator with random-selected seed

Returns

integer pseudo random value

Here is the call graph for this function:



4.6 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** ()

4.6.1 Detailed Description

Double linked list functions

Date

Created on: Apr 6, 2011

Author

iizke

4.7 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** ()

4.7.1 Detailed Description

Double Linked List structure

Date

Created on: Apr 6, 2011

Author

iizke

4.7.2 Define Documentation

4.7.2.1 #define container_of(ptr, type, member)

Value:

```
({           \
    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.

4.7.3 Typedef Documentation

4.7.3.1 typedef struct linked_list LINKED_LIST

Linked list structure

4.7.3.2 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)

4.8 matrix/matrix.c File Reference

```
#include "../error.h"
#include "matrix.h"
```

Functions

- int **matrix_get_row** (MATRIX *m, int row, ROW **r)
- int **matrix_init** (MATRIX **m)
- int **matrix_setup** (FILE *f)
- int **matrix_build_index** (MATRIX *m, int type)
- void * **matrix_get_value** (MATRIX *m, int row, int col)

4.8.1 Detailed Description

Date

Created on: May 25, 2011

Author

iizke

4.9 matrix/matrix.h File Reference

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

Data Structures

- struct **edge**
- struct **row**
- struct **matrix**

Typedefs

- typedef struct **edge** **EDGE**
- typedef struct **row** **ROW**
- typedef struct **matrix** **MATRIX**

Functions

- int **matrix_get_row** (MATRIX *, int row, ROW **)
- int **matrix_init** (MATRIX **)
- int **matrix_setup** (FILE *)
- int **matrix_build_index** (MATRIX *, int)
- void * **matrix_get_value** (MATRIX *, int, int)

4.9.1 Detailed Description

Date

Created on: May 25, 2011

Author

iizke

4.10 netlib.c File Reference

```
#include <stdio.h>
#include "error.h"
#include "netsim/netsim.h"
```

Functions

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

4.10.1 Detailed Description

Date

Created on: May 24, 2011

Author

iizke

4.11 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**

4.11.1 Detailed Description

Some functions related to config structure

Date

Created on: May 24, 2011

Author

iizke

4.11.2 Function Documentation**4.11.2.1 int config_init (CONFIG * *conf*)**

Initialize the structure of CONFIG

Parameters

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

Returns

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

4.11.2.2 int config_parse_file (char * *f*)

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

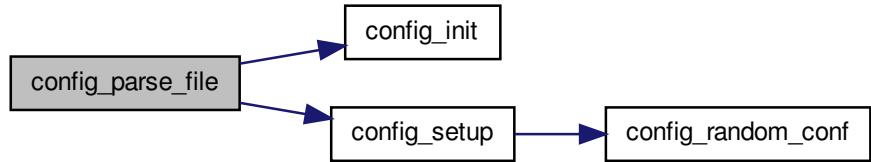
Parameters

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

Returns

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

Here is the call graph for this function:



4.11.2.3 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](#))

4.11.2.4 int config_setup (CONFIG * *conf*)

Configure random distribution in user configuration

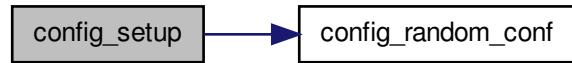
Parameters

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

Returns

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

Here is the call graph for this function:



4.12 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)`

4.12.1 Detailed Description

Define user configurations

Date

Created on: Apr 16, 2011

Author

iizke

4.12.2 Typedef Documentation

4.12.2.1 `typedef struct config CONFIG`

Configuration from user

4.12.2.2 `typedef struct queue_config QUEUE_CONF`

Queue configuration

4.12.2.3 `typedef struct random_config RANDOM_CONF`

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

4.12.2.4 `typedef struct stop_config STOP_CONF`

Define conditions of stopping program

4.12.3 Function Documentation

4.12.3.1 `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](#))

4.12.3.2 int config_parse_file (char * *f*)

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

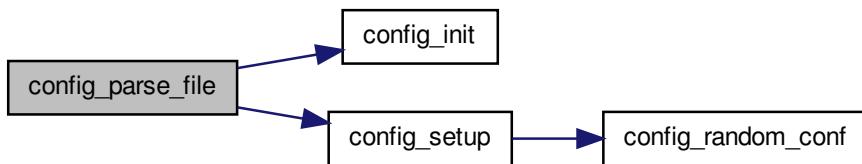
Parameters

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

Returns

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

Here is the call graph for this function:

**4.12.3.3 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](#))

4.12.3.4 int config_setup (CONFIG * *conf*)

Configure random distribution in user configuration

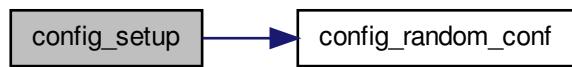
Parameters

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

Returns

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

Here is the call graph for this function:



4.13 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)

4.13.1 Detailed Description

Model of CSMA system (Carrier Sense Multiple Access)

Date

Created on: May 16, 2011

Author

iizke

4.13.2 Function Documentation

4.13.2.1 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:



4.13.2.2 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)

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

Generate arrival event

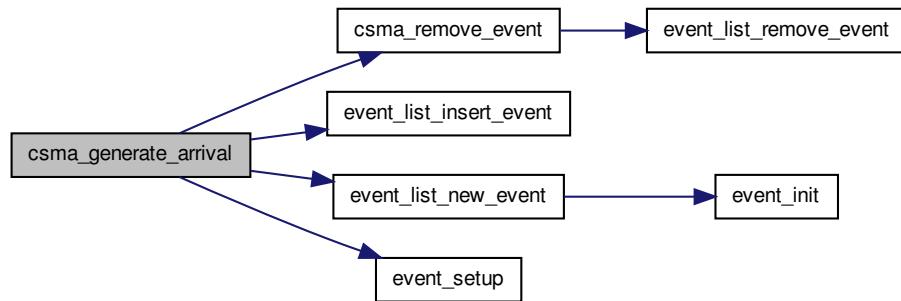
Parameters

<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:



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

Generate collision event

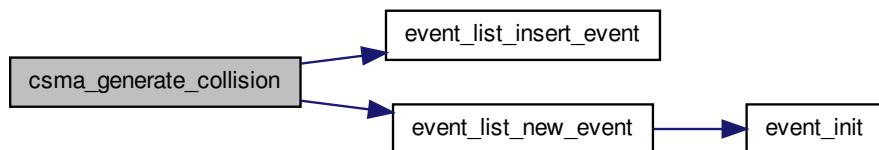
Parameters

<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:



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

Generate new end-service event.

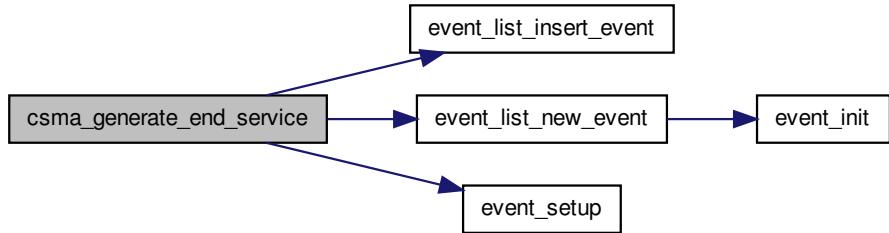
Parameters

<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:



4.13.2.6 EVENT* csma_generate_event (int *type*, PACKET * *p*, CONFIG * *conf*, SYS_STATE_OPS * *ops*)

Generate new event.

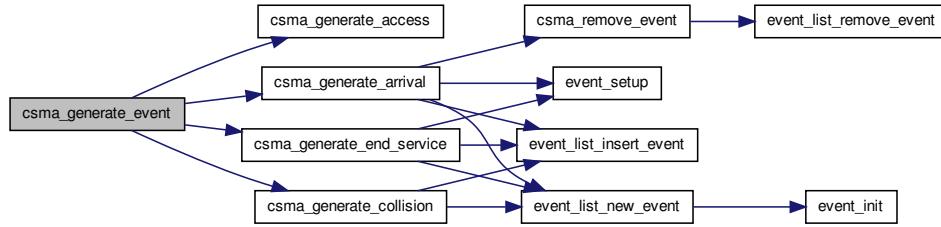
Parameters

<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:



4.13.2.7 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:



4.13.2.8 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:



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

Process Arrival event

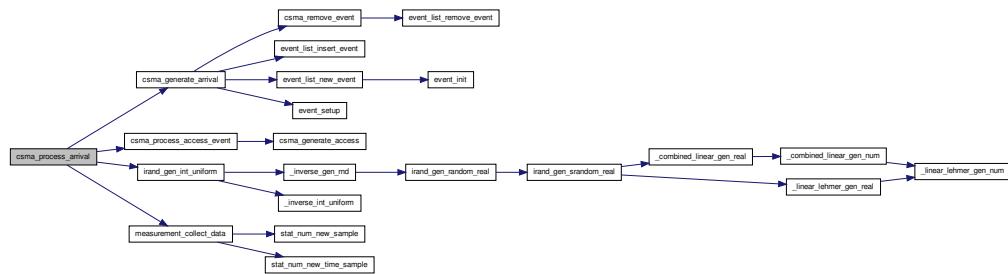
Parameters

<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:



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

Process Collision event

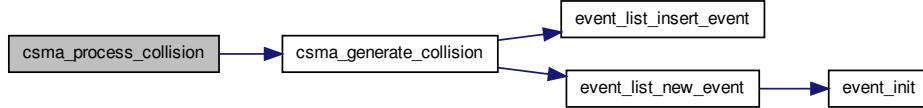
Parameters

<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:



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

Process End-service event

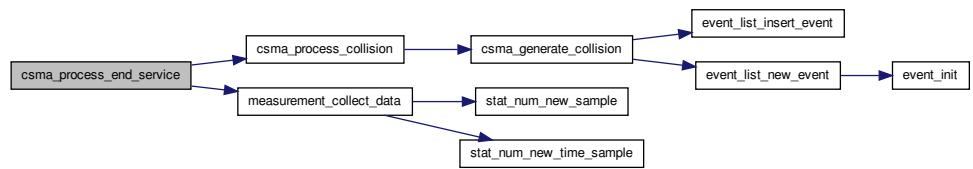
Parameters

<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:



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

Process an event.

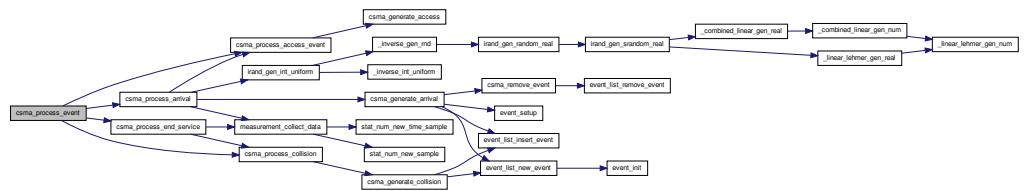
Parameters

<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:



4.13.2.13 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:

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

Initialize CSMA system state

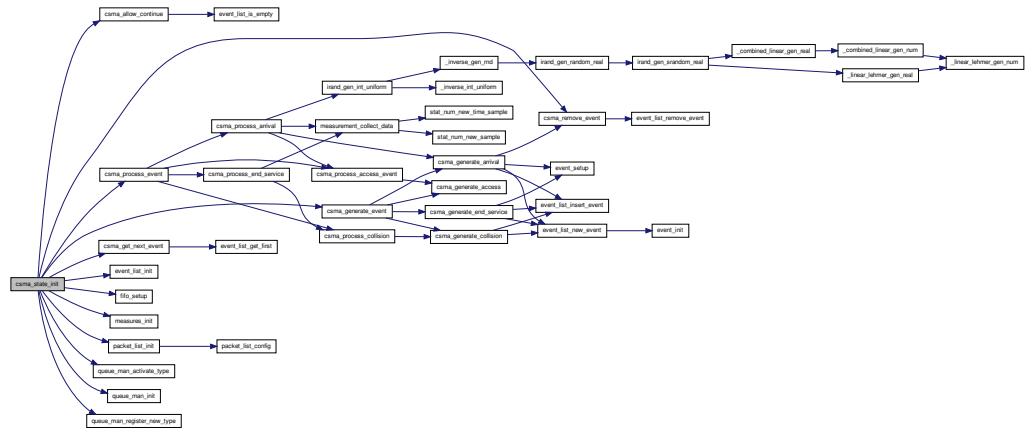
Parameters

<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:



4.14 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_TRANSFERING** 2
Queue is transfering packet.

Typedefs

- typedef struct **csma_state** CSMA_STATE

Functions

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

4.14.1 Detailed Description

CSMA system structure

Date

Created on: May 20, 2011

Author

iizke

4.14.2 Function Documentation

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

Initialize CSMA system state

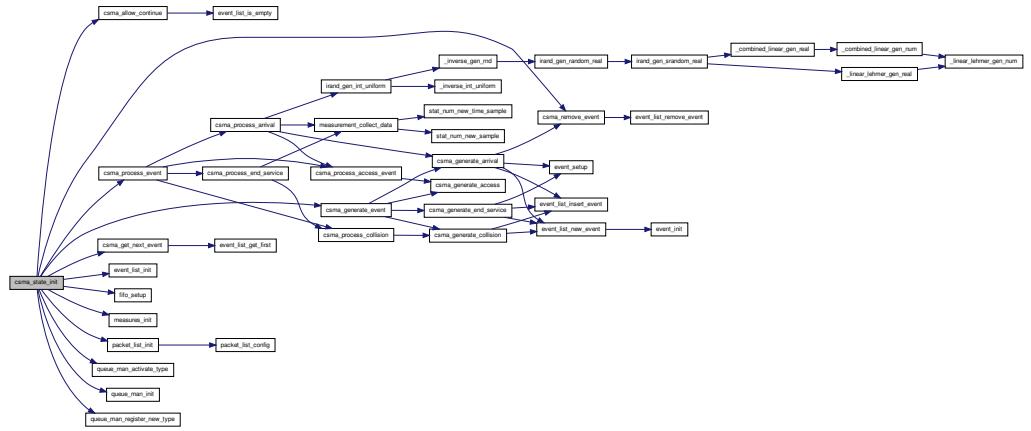
Parameters

<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:



4.15 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](#) ()

4.15.1 Detailed Description

Event and Event-list structure and related functions

Date

Created on: Apr 6, 2011

Author

iizke

4.15.2 Function Documentation**4.15.2.1 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)

4.15.2.2 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)

4.15.2.3 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)

4.15.2.4 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

4.15.2.5 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.

4.15.2.6 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:



4.15.2.7 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)

4.15.2.8 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)

4.15.2.9 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)

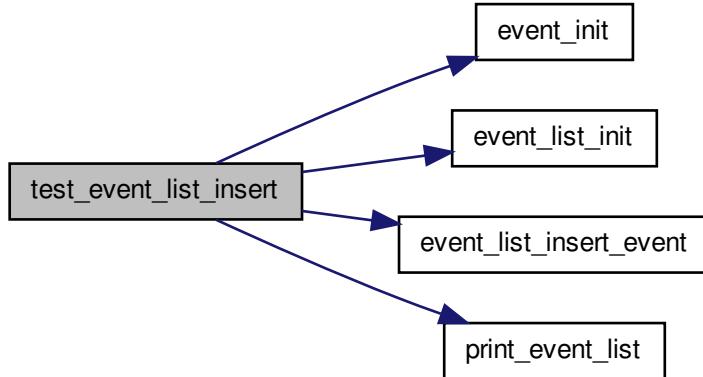
4.15.2.10 int test_event_list_insert ()

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:



4.16 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 *)
- int test_event_list_insert ()

4.16.1 Detailed Description

Event structure

Date

Created on: Apr 6, 2011

Author

iizke

4.16.2 Define Documentation

4.16.2.1 #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

4.16.3 Typedef Documentation

4.16.3.1 typedef struct event EVENT

Event structure

4.16.3.2 typedef struct event_list EVENT_LIST

Event list structure

4.16.3.3 typedef struct event_info EVENTINFO

Information in an event

4.16.4 Function Documentation

4.16.4.1 int event_init (EVENT * e)

Initialize parameters in Event structure

Parameters

e	: Event
---	---------

Returns

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

4.16.4.2 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](#))

4.16.4.3 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)

4.16.4.4 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

4.16.4.5 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.

4.16.4.6 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:

**4.16.4.7 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

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

Returns

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

4.16.4.8 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](#))

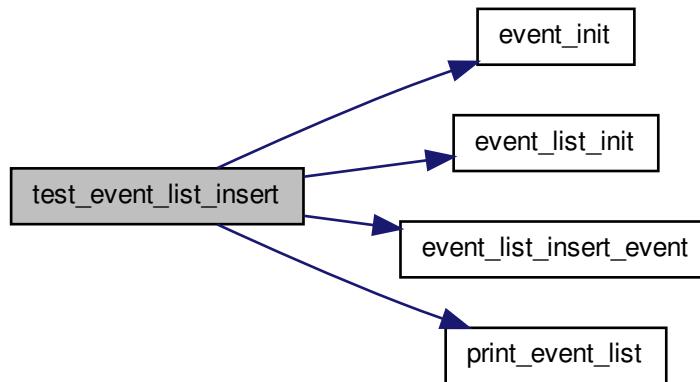
4.16.4.9 int test_event_list_insert()

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:

**4.17 netsim/netsim.c File Reference**

```
#include <stdlib.h>
#include <signal.h>
#include <string.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*

4.17.1 Detailed Description

Network simulator - the main framework.

Date

Created on: Apr 6, 2011

Author

iizke

4.17.2 Function Documentation

4.17.2.1 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](#))

4.17.2.2 int netsim_start (char * *conf_file*)

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

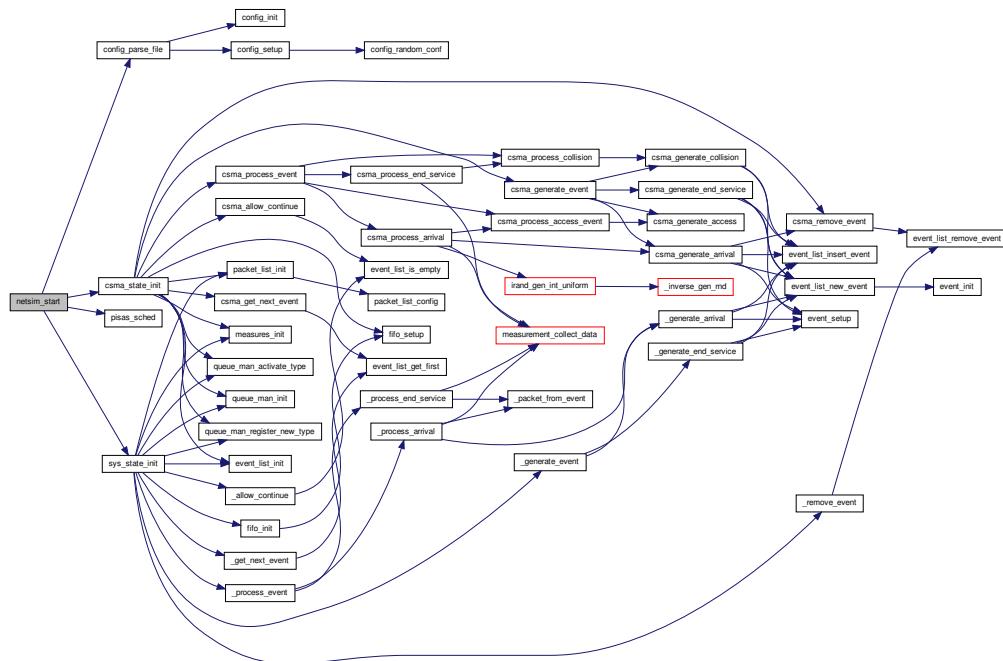
Parameters

<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:



4.17.2.3 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](#))

4.18 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*)

4.18.1 Detailed Description**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
----------	---------

fc	: random configuration
$curr_time$: 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

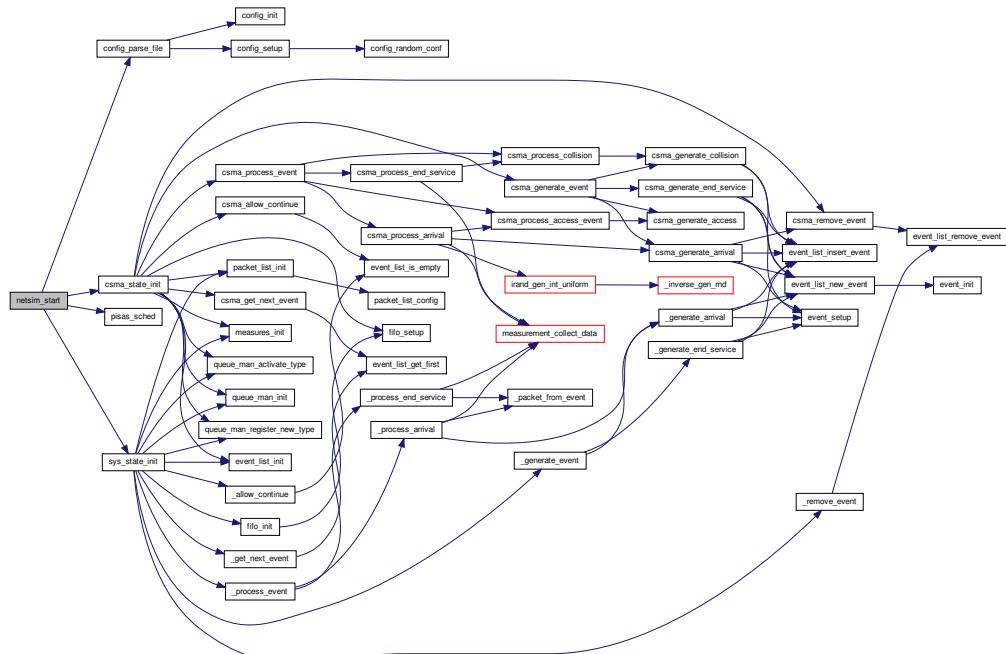
Parameters

<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:



4.19 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)

4.19.1 Detailed Description

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

Date

Created on: May 19, 2011

Author

iizke

4.19.2 Function Documentation

4.19.2.1 int `_allow_continue` (CONFIG * *conf*, SYS_STATE_OPS * *ops*)

Check whether system should be stopped or not.

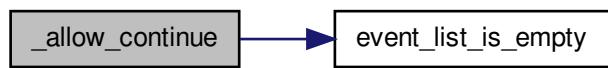
Parameters

<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:

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

Generate arrival event

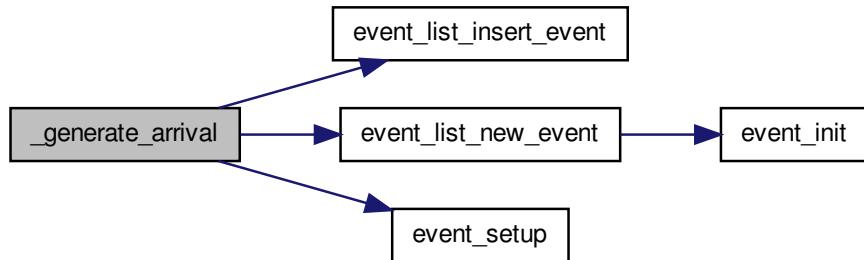
Parameters

<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:



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

Generate new end-service event.

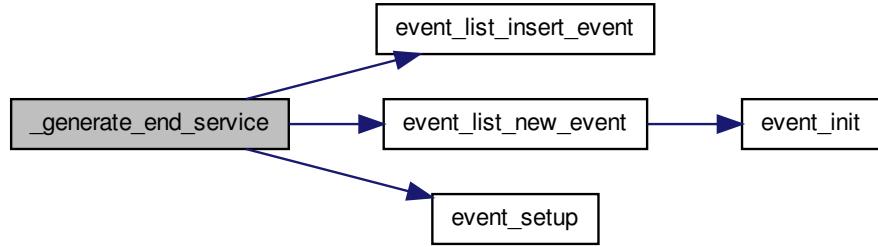
Parameters

<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:



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

Generate new event.

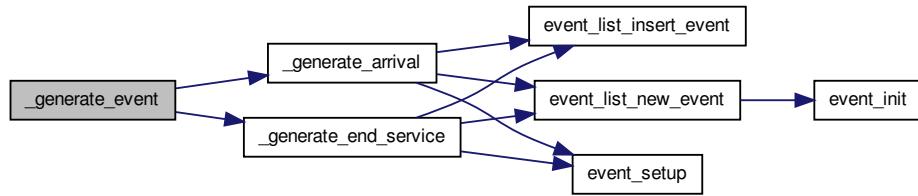
Parameters

<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:



4.19.2.5 int _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:



4.19.2.6 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](#))

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

Process Arrival event

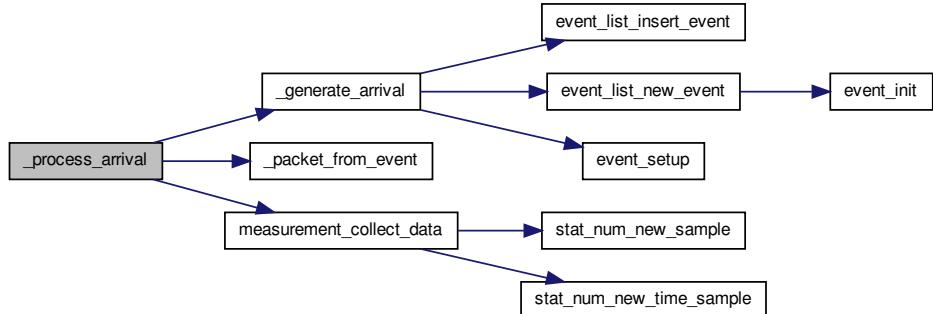
Parameters

<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:

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

Process end-service event

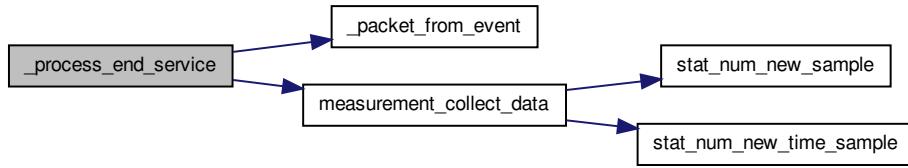
Parameters

<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:



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

Process an event.

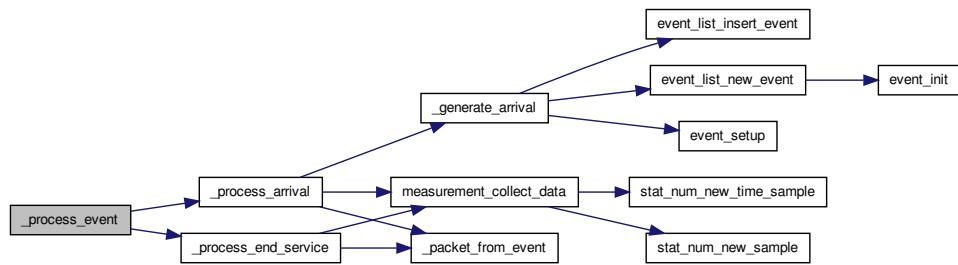
Parameters

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

Returns

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

Here is the call graph for this function:



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

Remove an event out of event list

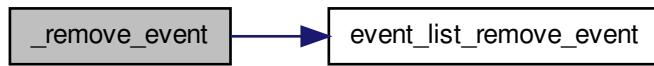
Parameters

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

Returns

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

Here is the call graph for this function:



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

Initialize system state of one-queue system

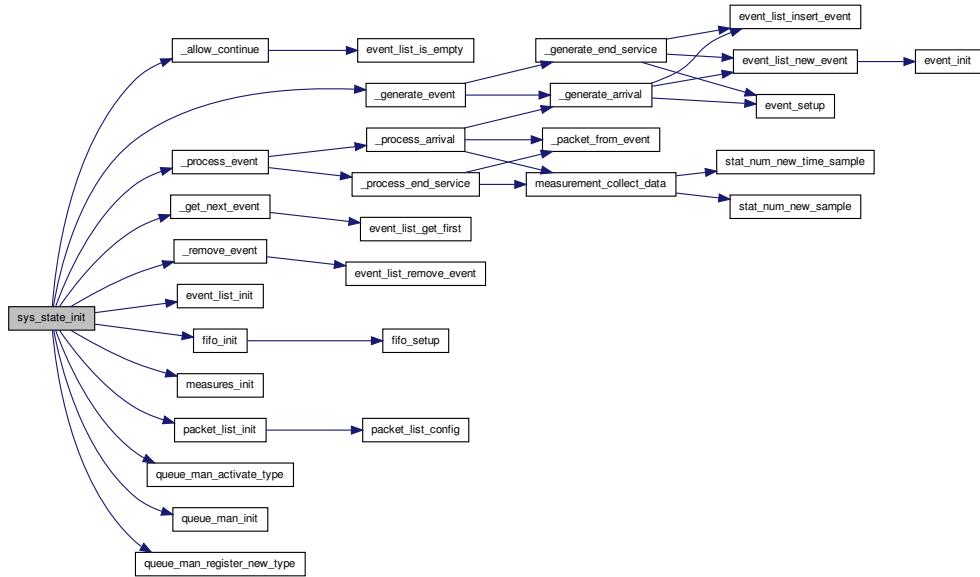
Parameters

<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:



4.20 netsim/sys_aqueue.h File Reference

```

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

4.20.1 Detailed Description

Specification of state in system with one queue

Date

Created on: May 19, 2011

Author

iizke

4.20.2 Typedef Documentation

4.20.2.1 `typedef struct sys_state SYS_STATE`

Structure representing the system state in simulation.

4.20.3 Function Documentation

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

Initialize system state of one-queue system

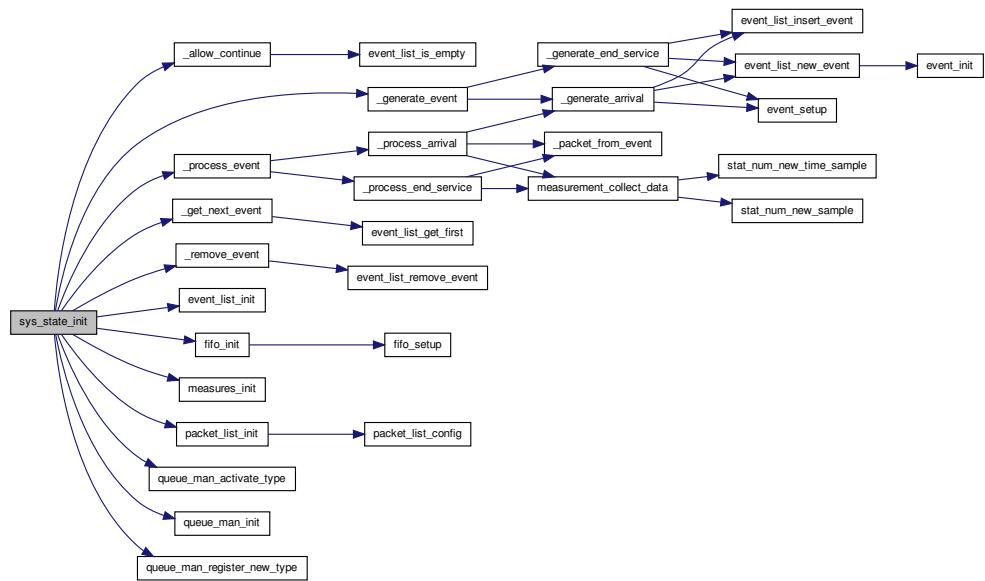
Parameters

<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:



4.21 network.c File Reference

```
#include "error.h"
#include "network.h"
```

Functions

- void * **network_find_shortest_path** (**NETWORK** *net, int alg, int src, int dest)

4.21.1 Detailed Description

Date

Created on: May 25, 2011

Author

iizke

4.22 network.h File Reference

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

Data Structures

- struct [shortest_path_dijkstra](#)
- struct [network](#)

Defines

- #define **NETWORK_SP_DIJKSTRA** 1
- #define **NETWORK_SP_BELLFORD** 2

Typedefs

- typedef struct [shortest_path_dijkstra](#) **SP_DIJKSTRA**
- typedef [LINKED_LIST_MAN](#) **SP_DIJKSTRA_LIST**
- typedef struct [network](#) **NETWORK**

Functions

- void * [network_find_shortest_path](#) (**NETWORK** *, int, int, int)

4.22.1 Detailed Description

Network structure: traffic, cost, ...

Date

Created on: May 25, 2011

Author

iizke

4.23 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)])

4.23.1 Detailed Description

Support quantile of essential distributions

Date

Created on: May 26, 2011

Author

iizke

4.23.2 Define Documentation

4.23.2.1 #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)) \
))))))))
```

4.23.2.2 #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)) \
))))
```

4.24 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)

4.24.1 Detailed Description

FIFO queue implementation.

Date

Created on: Apr 16, 2011

Author

iizke

4.24.2 Function Documentation

4.24.2.1 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:



4.24.2.2 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](#))

4.25 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)

4.25.1 Detailed Description

FIFO queue structure

Date

Created on: Apr 16, 2011

Author

iizke

4.25.2 Typedef Documentation

4.25.2.1 `typedef struct fifo_queue FIFO_QINFO`

FIFO queue structure

4.25.3 Function Documentation

4.25.3.1 `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:



4.25.3.2 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](#))

4.26 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)

4.26.1 Detailed Description

Measurement functions

Date

Created on: Apr 9, 2011

Author

iizke

4.26.2 Define Documentation

4.26.2.1 `#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)); }
```

4.26.3 Function Documentation

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

Collect new data into measurement structure

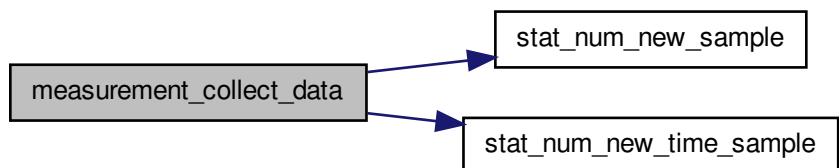
Parameters

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

Returns

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

Here is the call graph for this function:



4.26.3.2 int measures_init (MEASURES * m)

Initialization of structure MEASURES

Parameters

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

Returns

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

4.26.3.3 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](#))

4.27 queues/measures.h File Reference

```
#include "../def.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)

4.27.1 Detailed Description

Collect statistical information of queueing system

Date

Created on: Apr 6, 2011

Author

iizke

4.27.2 Typedef Documentation**4.27.2.1 `typedef struct measures MEASURES`**

MEASURE structure used to store some results when simulating queue system

4.27.3 Function Documentation**4.27.3.1 `int measurement_collect_data (MEASURES * m, PACKET * p, TIME curr_time)`**

Collect new data into measurement structure

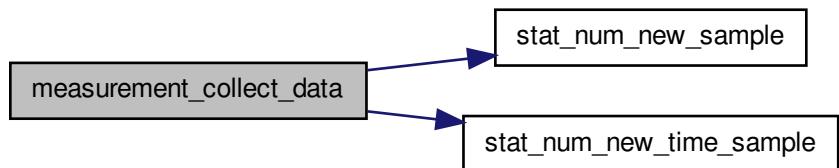
Parameters

<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:



4.27.3.2 int measures_init (MEASURES * m)

Initialization of structure MEASURES

Parameters

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

Returns

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

4.27.3.3 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](#))

4.28 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 \(PACKET *p\)](#)

4.28.1 Detailed Description

Packet functions

Date

Created on: Apr 8, 2011

Author

iizke

4.28.2 Function Documentation

4.28.2.1 int measurement_self_collect_data (PACKET * p)

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

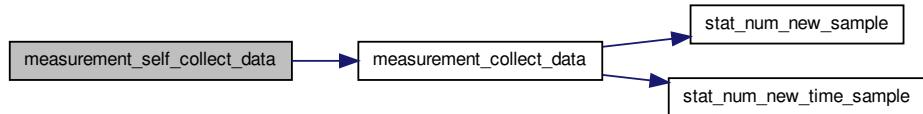
Parameters

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

Returns

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

Here is the call graph for this function:



4.28.2.2 int packet_init (PACKET * p)

Initialization a packet

Parameters

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

Returns

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

4.28.2.3 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](#))

4.28.2.4 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](#))

4.28.2.5 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:



4.28.2.6 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](#))

4.28.2.7 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.

4.28.2.8 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:

**4.28.2.9 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](#))

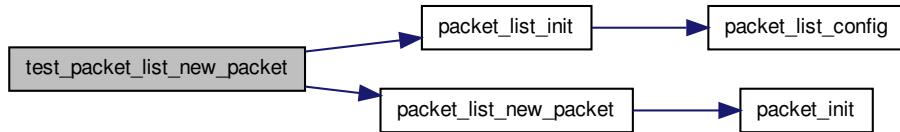
4.28.2.10 int test_packet_list_new_packet()

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:



4.29 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)

4.29.1 Detailed Description

Packet structure

Date

Created on: Apr 8, 2011

Author

iizke

4.29.2 Typedef Documentation

4.29.2.1 `typedef struct packet PACKET`

Packet structure

4.29.2.2 `typedef struct packet_info PACKET_INFO`

Packet information

4.29.2.3 `typedef struct packet_list PACKET_LIST`

Packet list structure

4.29.3 Function Documentation

4.29.3.1 `int measurement_self_collect_data(PACKET * p)`

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

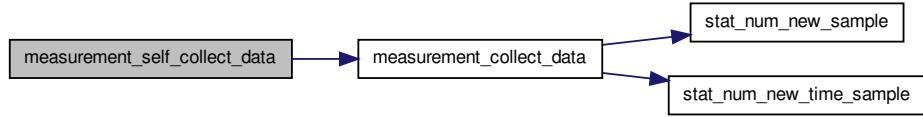
Parameters

<code>p</code>	: Packet
----------------	----------

Returns

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

Here is the call graph for this function:



4.29.3.2 int packet_init (PACKET * p)

Initialization a packet

Parameters

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

Returns

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

4.29.3.3 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](#))

4.29.3.4 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](#))

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

Initialization of a packet list

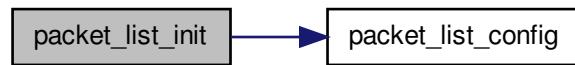
Parameters

<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:

**4.29.3.6 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](#))

4.29.3.7 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.

4.29.3.8 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:

**4.29.3.9 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](#))

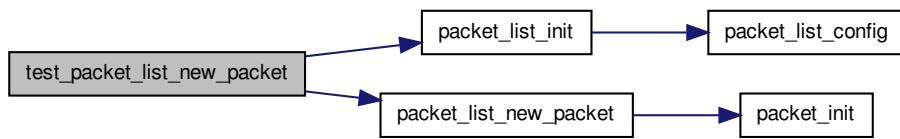
4.29.3.10 int test_packet_list_new_packet()

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:



4.30 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\(QUEUE_MAN *qm, int type\)](#)

4.30.1 Detailed Description

Implementation of essential queue operations.

Date

Created on: Apr 16, 2011

Author

iizke

4.30.2 Function Documentation

4.30.2.1 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)

4.30.2.2 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)

4.30.2.3 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)

4.30.2.4 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](#))

4.31 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)

4.31.1 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

4.31.2 Typedef Documentation

4.31.2.1 `typedef struct queue_type_list QUEUE_MAN`

Queue manager structure

4.31.3 Function Documentation

4.31.3.1 `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)

4.31.3.2 `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)

4.31.3.3 `int queue_man_register_new_type (QUEUE_MAN * qm, QUEUE_TYPE * qi)`

Register new queue-type into queue-manager

Parameters

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

Returns

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

4.31.3.4 `int queue_man_unregister_type (QUEUE_MAN *qm, QUEUE_TYPE *qi)`

Remove a queue type out of queue-manager

Parameters

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

Returns

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

4.32 random.h File Reference

```
#include <stdio.h>
#include "irand/irand.h"
#include "ranlib/ranlib.h"
```

Defines

- `#define LIB_RANDOM_IRAND 0`
Random library implemented by iizke.
- `#define LIB_RANDOM_RANLIB 1`
Random library implemented by ranlib.
- `#define LIB_RANDOM_POLIRAND 2`
Random library implemented by polito-er.
- `#define random_dist_gen(rdist) ((rdist)->gen(rdist))`
- `#define random_dist_cdf(rdist, value) (rdist->cdf(rdist, value))`

Functions

- `int random_init ()`
- `float gen_uniform (float from, float to)`
- `long gen_int_uniform (long from, long to)`
- `int gen_bernoulli (double p)`
- `float gen_exponential (float rate)`
- `float gen_normal (float mean, float sd)`
- `long gen_poisson (float lambda)`
- `int random_dist_init_uniform1 (RANDOM_DIST *rd, double from, double to)`
- `int random_dist_init_uniform0 (RANDOM_DIST *rd, struct uniform_params *up)`

- int `random_dist_init_exp0` (RANDOM_DIST *rd, double *lambda)
- int `random_dist_init_exp1` (RANDOM_DIST *rd, double lambda)
- int `random_dist_init_file0` (RANDOM_DIST *rd, FILE *f)
- int `random_dist_init_bernoulli0` (RANDOM_DIST *rd, double *prob)
- int `random_dist_init_bernoulli1` (RANDOM_DIST *rd, double prob)

4.32.1 Detailed Description

Random generator and distribution structure

Date

Created on: Apr 6, 2011

Author

iizke

4.32.2 Function Documentation

4.32.2.1 int `gen_bernoulli` (double *p*)

Generate a real number following Bernoulli distribution

Parameters

<i>p</i>	: Probability of success event (return 1)
----------	---

Returns

0 or 1

Here is the call graph for this function:



4.32.2.2 float `gen_exponential` (float *lambda*)

Generate a real number following exponential distribution

Parameters

<i>lambda</i>	: lambda
---------------	----------

Returns

a real number

Here is the call graph for this function:

**4.32.2.3 long gen_int_uniform (long from, long to)**

Generate a integer number following uniform distribution

Parameters

<i>from,to</i>	: range of generated value
----------------	----------------------------

Returns

an integer number in range

Here is the call graph for this function:

**4.32.2.4 float gen_normal (float mean, float sd)**

Generate a real number following normal distribution

Parameters

<i>mean</i>	: mean
<i>sd</i>	: standard deviation

Returns

a real number

4.32.2.5 long gen_poisson (float lambda)

Generate a real number following Poisson distribution

Parameters

<i>lambda</i>	: lambda
---------------	----------

Returns

an integer number

4.32.2.6 float gen_uniform (float from, float to)

Generate a real number following uniform distribution

Parameters

<i>from,to</i>	: range of generated value
----------------	----------------------------

Returns

a real number in range

Here is the call graph for this function:



4.32.2.7 int random_dist_init_bernoulli0 (RANDOM_DIST * rd, double * prob)

Initialize random distribution structure of bernoulli distribution

Parameters

<i>rd</i>	: Random distribution structure
<i>prob</i>	: pointer of probability of success value (value 1)

Returns

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

4.32.2.8 int random_dist_init_bernoulli1 (RANDOM_DIST * rd, double prob)

Initialize random distribution structure of bernoulli distribution

Parameters

<i>rd</i>	: Random distribution structure
<i>prob</i>	: probability of success value (value 1)

Returns

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

4.32.2.9 int random_dist_init_exp0 (RANDOM_DIST * *rd*, double * *lambda*)

Initialize random distribution structure of exponential distribution

Parameters

<i>rd</i>	: Random distribution structure
<i>lambda</i>	: pointer of lambda value

Returns

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

4.32.2.10 int random_dist_init_exp1 (RANDOM_DIST * *rd*, double *lambda*)

Initialize random distribution structure of exponential distribution

Parameters

<i>rd</i>	: Random distribution structure
<i>lambda</i>	: lambda value

Returns

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

4.32.2.11 int random_dist_init_file0 (RANDOM_DIST * *rd*, FILE * *f*)

Initialize random structure extracted from file

Parameters

<i>rd</i>	: Random distribution structure
<i>f</i>	: pointer to a file

Returns

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

4.32.2.12 int random_dist_init_uniform0 (RANDOM_DIST * rd, struct uniform_params * up)

Initialize random distribution structure of uniform distribution

Parameters

<i>rd</i>	: Random distribution structure
<i>up</i>	: structure containing upper bound and lower bound of uniform distribution

Returns

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

4.32.2.13 int random_dist_init_uniform1 (RANDOM_DIST * rd, double from, double to)

Initialize random distribution structure of uniform distribution

Parameters

<i>rd</i>	: Random distribution structure
<i>from</i>	: Lower bound value
<i>to</i>	: Upper bound value

Returns

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

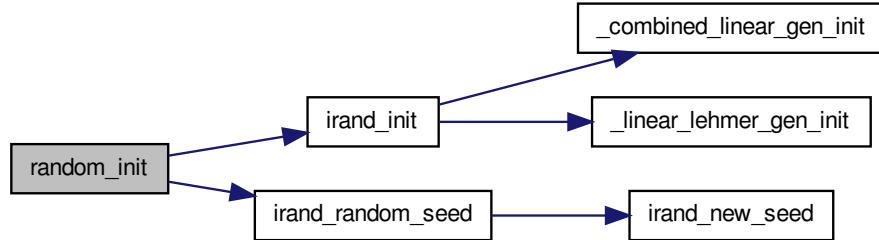
4.32.2.14 int random_init ()

Initialize some parameters of random generators

Returns

SUCCESS

Here is the call graph for this function:



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

- float **calc_avg** (float last_avg, int nsamples, float newsample)
- 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 float calc_avg (float last_avg, int nsamples, float newsample)

Calculate the average value incrementally

Parameters

<i>last_avg</i>	: Last average value
<i>nsamples</i>	: Last number of samples
<i>newsample</i>	: New sample value

Returns

a real number

4.33.2.2 int stat_num_init (STAT_NUM * *m*)

Initialization of STAT_NUM structure (statistical information)

Parameters

<i>m</i>	: STAT_NUM
----------	------------

Returns

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

4.33.2.3 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

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

Returns

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

4.33.2.4 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 continous 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 TYPEDOC 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, 108
_combined_linear_gen_init
 rndnum.c, 66
_combined_linear_gen_num
 rndnum.c, 66
_combined_linear_gen_real
 rndnum.c, 67
_composition_gen_rnd
 rnndist.c, 58
_convolution_gen_rnd
 rnndist.c, 58
_generate_arrival
 sys_aqueue.c, 109
_generate_end_service
 sys_aqueue.c, 109
_generate_event
 sys_aqueue.c, 110
_get_next_event
 sys_aqueue.c, 111
_inverse_bernoulli
 rnndist.c, 59
_inverse_empirical
 rnndist.c, 59
_inverse_exp
 rnndist.c, 59
_inverse_gen_rnd
 rnndist.c, 60
_inverse_geometric
 rnndist.c, 60
_inverse_int_uniform
 rnndist.c, 60
_inverse_pareto
 rnndist.c, 61
_inverse_uniform
 rnndist.c, 61
_inverse_weibull
 rnndist.c, 61
_linear_lehmer_gen_init
 rndnum.c, 67
_linear_lehmer_gen_num
 rndnum.c, 68
_linear_lehmer_gen_real
 rndnum.c, 68
_packet_from_event
 sys_aqueue.c, 111
_process_arrival
 sys_aqueue.c, 112
_process_end_service
 sys_aqueue.c, 112
_process_event
 sys_aqueue.c, 113
_remove_event
 sys_aqueue.c, 113

calc_avg
 stat_num.c, 148
check_null_pointer
 error.h, 49
chisqr.c
 test_chisqr, 151
combined_linear_congruential_gen, 5
COMBINED_LINEAR_GEN
 rndnum.c, 66
CONFIG
 config.h, 80
config, 6
 runtime_state, 7
config.c
 config_init, 77
 config_parse_file, 77
 config_random_conf, 78
 config_setup, 78
config.h
 CONFIG, 80
 config_init, 80
 config_parse_file, 81
 config_random_conf, 81
 config_setup, 81
 QUEUE_CONF, 80
 RANDOM_CONF, 80
 STOP_CONF, 80

config_init
 config.c, 77
 config.h, 80
config_parse_file
 config.c, 77
 config.h, 81
config_random_conf
 config.c, 78
 config.h, 81
config_setup
 config.c, 78
 config.h, 81
container_of
 linked_list.h, 74
csma.c
 csma_allow_continue, 83
 csma_generate_access, 84
 csma_generate_arrival, 84
 csma_generate_collision, 85
 csma_generate_end_service, 85
 csma_generate_event, 86
 csma_get_next_event, 87
 csma_process_access_event, 87
 csma_process_arrival, 88
 csma_process_collision, 89
 csma_process_end_service, 89
 csma_process_event, 90
 csma_remove_event, 90
 csma_state_init, 91
csma.h
 csma_state_init, 93
csma_allow_continue
 csma.c, 83
csma_conf, 8
csma_generate_access
 csma.c, 84
csma_generate_arrival
 csma.c, 84
csma_generate_collision
 csma.c, 85
csma_generate_end_service
 csma.c, 85
csma_generate_event
 csma.c, 86
csma_get_next_event
 csma.c, 87
csma_process_access_event
 csma.c, 87
csma_process_arrival
 csma.c, 88
 csma_process_collision
 csma.c, 89
 csma_process_end_service
 csma.c, 89
 csma_process_event
 csma.c, 90
 csma_remove_event
 csma.c, 90
 csma_state, 9
 csma_state_init
 csma.c, 91
 csma.h, 93
 def.h, 47
 TIME, 47
edge, 10
empirical_params, 10
error.h, 47
 check_null_pointer, 49
 iprintf, 49
 try, 49
EVENT
 event.h, 100
event, 11
event.c
 event_init, 95
 event_list_get_first, 95
 event_list_init, 95
 event_list_insert_event, 95
 event_list_is_empty, 96
 event_list_new_event, 96
 event_list_remove_event, 96
 event_save, 97
 print_event_list, 97
 test_event_list_insert, 97
event.h
 EVENT, 100
 event_init, 100
 EVENT_LIST, 100
 event_list_get_first, 100
 event_list_init, 101
 event_list_insert_event, 101
 event_list_is_empty, 101
 event_list_new_event, 102
 event_list_remove_event, 102
 event_save, 102
 EVENTINFO, 100
 swap_prev_event, 100
 test_event_list_insert, 103

event_info, 13
 event_init
 event.c, 95
 event.h, 100
 EVENT_LIST
 event.h, 100
 event_list, 15
 event_list_get_first
 event.c, 95
 event.h, 100
 event_list_init
 event.c, 95
 event.h, 101
 event_list_insert_event
 event.c, 95
 event.h, 101
 event_list_is_empty
 event.c, 96
 event.h, 101
 event_list_new_event
 event.c, 96
 event.h, 102
 event_list_remove_event
 event.c, 96
 event.h, 102
 event_save
 event.c, 97
 event.h, 102
 event_setup
 netsim.c, 104
 netsim.h, 106
 EVENTINFO
 event.h, 100

 fifo.c
 fifo_init, 120
 fifo_setup, 121
 fifo.h
 fifo_init, 122
 FIFO_QINFO, 122
 fifo_setup, 122
 fifo_init
 fifo.c, 120
 fifo.h, 122
 FIFO_QINFO
 fifo.h, 122
 fifo_queue, 16
 fifo_setup
 fifo.c, 121
 fifo.h, 122
 gen_bernoulli
 random.h, 142
 gen_exponential
 random.h, 142
 gen_int_uniform
 random.h, 143
 gen_normal
 random.h, 143
 gen_poisson
 random.h, 143
 gen_uniform
 random.h, 144
 iprintf
 error.h, 49
 irand.h
 irand_gen_bernoulli, 51
 irand_gen_erlang, 51
 irand_gen_exp, 52
 irand_gen_int_uniform, 52
 irand_gen_pareto, 53
 irand_gen_random, 53
 irand_gen_random_real, 54
 irand_gen_srandom, 54
 irand_gen_srandom_real, 55
 irand_gen_uniform, 55
 irand_init, 55
 irand_new_seed, 56
 irand_random_seed, 56
 irand/irand.h, 50
 irand/rnddist.c, 57
 irand/rndnum.c, 64
 irand_gen_bernoulli
 irand.h, 51
 rnddist.c, 62
 irand_gen_erlang
 irand.h, 51
 rnddist.c, 62
 irand_gen_exp
 irand.h, 52
 rnddist.c, 62
 irand_gen_int_uniform
 irand.h, 52
 rnddist.c, 63
 irand_gen_pareto
 irand.h, 53
 rnddist.c, 63
 irand_gen_random
 irand.h, 53
 rndnum.c, 68

irand_gen_random_real
 irand.h, 54
 rndnum.c, 69
irand_gen_srandom
 irand.h, 54
 rndnum.c, 69
irand_gen_srandom_real
 irand.h, 55
 rndnum.c, 70
irand_gen_uniform
 irand.h, 55
 rnndist.c, 64
irand_init
 irand.h, 55
 rndnum.c, 70
irand_new_seed
 irand.h, 56
 rndnum.c, 71
irand_random_seed
 irand.h, 56
 rndnum.c, 71

linear_congruential_gen, 18
LINEAR_LEHMER_GEN
 rndnum.c, 66
LINKED_LIST
 linked_list.h, 74
linked_list, 19
linked_list.h
 container_of, 74
 LINKED_LIST, 74
 LINKED_LIST_MAN, 74
LINKED_LIST_MAN
 linked_list.h, 74
linked_list_manager, 20
list/linked_list.c, 72
list/linked_list.h, 72

matrix, 21
matrix/matrix.c, 74
matrix/matrix.h, 75
measurement_collect_data
 measures.c, 124
 measures.h, 126
measurement_self_collect_data
 packet.c, 128
 packet.h, 133
MEASURES
 measures.h, 126
measures, 21

measures.c
 measurement_collect_data, 124
 measures_init, 124
 print_measurement, 125
 print_statistical_value, 124
measures.h
 measurement_collect_data, 126
 MEASURES, 126
 measures_init, 126
 print_measurement, 127
measures_init
 measures.c, 124
 measures.h, 126

netlib.c, 76
netsim.c
 event_setup, 104
 netsim_start, 104
 pisas_sched, 105
netsim.h
 event_setup, 106
 netsim_start, 107
netsim/conf/config.c, 76
netsim/conf/config.h, 79
netsim/csma.c, 82
netsim/csma.h, 92
netsim/event.c, 94
netsim/event.h, 98
netsim/netsim.c, 103
netsim/netsim.h, 106
netsim/sys_aqueue.c, 108
netsim/sys_aqueue.h, 115
netsim_start
 netsim.c, 104
 netsim.h, 107
network, 23
network.c, 117
network.h, 118

PACKET
 packet.h, 133
packet, 24
packet.c
 measurement_self_collect_data, 128
 packet_init, 128
 packet_list_config, 128
 packet_list_get_first, 129
 packet_list_init, 129
 packet_list_insert_packet, 130
 packet_list_is_empty, 130

packet_list_new_packet, 130
 packet_list_remove_packet, 131
 test_packet_list_new_packet, 131
packet.h
 measurement_self_collect_data, 133
 PACKET, 133
 PACKET_INFO, 133
 packet_init, 134
 PACKET_LIST, 133
 packet_list_config, 134
 packet_list_get_first, 134
 packet_list_init, 135
 packet_list_insert_packet, 135
 packet_list_is_empty, 135
 packet_list_new_packet, 136
 packet_list_remove_packet, 136
 test_packet_list_new_packet, 136
PACKET_INFO
 packet.h, 133
packet_info, 25
packet_init
 packet.c, 128
 packet.h, 134
PACKET_LIST
 packet.h, 133
packet_list, 26
packet_list_config
 packet.c, 128
 packet.h, 134
packet_list_get_first
 packet.c, 129
 packet.h, 134
packet_list_init
 packet.c, 129
 packet.h, 135
packet_list_insert_packet
 packet.c, 130
 packet.h, 135
packet_list_is_empty
 packet.c, 130
 packet.h, 135
packet_list_new_packet
 packet.c, 130
 packet.h, 136
packet_list_remove_packet
 packet.c, 131
 packet.h, 136
pisas_sched
 netsim.c, 105
print_event_list
 event.c, 97
 print_measurement
 measures.c, 125
 measures.h, 127
 print_statistical_value
 measures.c, 124
 pvalue_chi_id
 quantile.h, 119
 pvalue_normal_id
 quantile.h, 119
 quantile.h, 118
 pvalue_chi_id, 119
 pvalue_normal_id, 119
QUEUE_CONF
 config.h, 80
queue_config, 28
QUEUE_MAN
 queue_man.h, 140
queue_man.c
 queue_man_activate_type, 138
 queue_man_init, 138
 queue_man_register_new_type, 138
 queue_man_unregister_type, 138
queue_man.h
 QUEUE_MAN, 140
 queue_man_activate_type, 140
 queue_man_init, 140
 queue_man_register_new_type, 140
 queue_man_unregister_type, 140
queue_man_activate_type
 queue_man.c, 138
 queue_man.h, 140
queue_man_init
 queue_man.c, 138
 queue_man.h, 140
queue_man_register_new_type
 queue_man.c, 138
 queue_man.h, 140
queue_man_unregister_type
 queue_man.c, 138
 queue_man.h, 140
queue_type, 28
queue_type_list, 30
queues/fifo.c, 119
queues/fifo.h, 121
queues/measures.c, 123
queues/measures.h, 125
queues/packet.c, 127
queues/packet.h, 132

queues/queue_man.c, 137
queues/queue_man.h, 139

random.h, 141

- gen_bernoulli, 142
- gen_exponential, 142
- gen_int_uniform, 143
- gen_normal, 143
- gen_poisson, 143
- gen_uniform, 144
- random_dist_init_bernoulli0, 144
- random_dist_init_bernoulli1, 144
- random_dist_init_exp0, 145
- random_dist_init_exp1, 145
- random_dist_init_file0, 145
- random_dist_init_uniform0, 145
- random_dist_init_uniform1, 146
- random_init, 146

RANDOM_CONF

- config.h, 80

random_config, 31

random_dist_init_bernoulli

- random.h, 144

random_dist_init_bernoulli1

- random.h, 144

random_dist_init_exp0

- random.h, 145

random_dist_init_exp1

- random.h, 145

random_dist_init_file0

- random.h, 145

random_dist_init_uniform0

- random.h, 145

random_dist_init_uniform1

- random.h, 146

random_distribution, 32

random_init

- random.h, 146

rnndist.c

- _composition_gen_rnd, 58
- _convolution_gen_rnd, 58
- _inverse_bernoulli, 59
- _inverse_empirical, 59
- _inverse_exp, 59
- _inverse_gen_rnd, 60
- _inverse_geometric, 60
- _inverse_int_uniform, 60
- _inverse_pareto, 61
- _inverse_uniform, 61
- _inverse_weibull, 61

irand_gen_bernoulli, 62
irand_gen_erlang, 62
irand_gen_exp, 62
irand_gen_int_uniform, 63
irand_gen_pareto, 63
irand_gen_uniform, 64

rndnum.c

- _combined_linear_gen_init, 66
- _combined_linear_gen_num, 66
- _combined_linear_gen_real, 67
- _linear_lehmer_gen_init, 67
- _linear_lehmer_gen_num, 68
- _linear_lehmer_gen_real, 68
- COMBINED_LINEAR_GEN, 66
- irand_gen_random, 68
- irand_gen_random_real, 69
- irand_gen_random, 69
- irand_gen_random_real, 70
- irand_init, 70
- irand_new_seed, 71
- irand_random_seed, 71
- LINEAR_LEHMER_GEN, 66

row, 33

runtime_state

- config, 7

shortest_path_dijkstra, 34

STAT_NUM

- stat_num.h, 150

stat_num.c, 147

- calc_avg, 148
- stat_num_init, 148
- stat_num_new_sample, 148
- stat_num_new_time_sample, 149

stat_num.h, 149

- STAT_NUM, 150
- stat_num_init, 150
- stat_num_new_sample, 150
- stat_num_new_time_sample, 150

stat_num_init

- stat_num.c, 148
- stat_num.h, 150

stat_num_new_sample

- stat_num.c, 148
- stat_num.h, 150

stat_num_new_time_sample

- stat_num.c, 149
- stat_num.h, 150

statistical_number, 34

STOP_CONF

config.h, 80
stop_config, 35
swap_prev_event
 event.h, 100
sys_aqueue.c
 _allow_continue, 108
 _generate_arrival, 109
 _generate_end_service, 109
 _generate_event, 110
 _get_next_event, 111
 _packet_from_event, 111
 _process_arrival, 112
 _process_end_service, 112
 _process_event, 113
 _remove_event, 113
 sys_state_init, 114
sys_aqueue.h
 SYS_STATE, 116
 sys_state_init, 116
SYS_STATE
 sys_aqueue.h, 116
sys_state, 36
sys_state_init
 sys_aqueue.c, 114
 sys_aqueue.h, 116
system_state_operations, 38

test_chisqr
 chisqr.c, 151
test_event_list_insert
 event.c, 97
 event.h, 103
test_packet_list_new_packet
 packet.c, 131
 packet.h, 136
test_params, 41
tests/chisqr.c, 151
TIME
 def.h, 47
time, 41
try
 error.h, 49

uniform_params, 42
weibull_params, 42

yy_bs_column
 yy_buffer_state, 43
yy_bs_lineno