

My Project

1.1

Generated by Doxygen 1.8.11

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	Cpu_List Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	data	5
3.1.2.2	next	5
3.2	Device_Collection Struct Reference	5
3.2.1	Detailed Description	6
3.2.2	Field Documentation	6
3.2.2.1	devices	6
3.2.2.2	next	6
3.2.2.3	prev	6
3.3	Interrupt_Collection Struct Reference	6
3.3.1	Field Documentation	6
3.3.1.1	interrupts	6
3.3.1.2	next	6
3.3.1.3	prev	6
3.4	Interrupt_Collection2 Struct Reference	7

3.4.1	Field Documentation	7
3.4.1.1	interrupts	7
3.4.1.2	next	7
3.4.1.3	prev	7
3.5	Mega_Data Struct Reference	7
3.5.1	Field Documentation	8
3.5.1.1	cpu_stats	8
3.5.1.2	device_list	8
3.5.1.3	interrupts_list	8
3.5.1.4	mem_stats	8
3.5.1.5	net_stats	8
3.5.1.6	next	8
3.5.1.7	task_list	8
3.6	NetMem_list Struct Reference	8
3.6.1	Detailed Description	8
3.6.2	Field Documentation	8
3.6.2.1	data	8
3.6.2.2	next	8
3.7	Task_Collection Struct Reference	9
3.7.1	Field Documentation	9
3.7.1.1	next	9
3.7.1.2	prev	9
3.7.1.3	task	9
3.8	Unification Union Reference	9
3.8.1	Detailed Description	9
3.8.2	Field Documentation	10
3.8.2.1	conformation	10
3.8.2.2	data_pack	10
3.8.2.3	devices	10
3.8.2.4	interrupts	10
3.8.2.5	interrupts_send	10
3.8.2.6	memory_usage	10
3.8.2.7	network	10
3.8.2.8	task	10

4 File Documentation	11
4.1 buttons.c File Reference	11
4.1.1 Function Documentation	11
4.1.1.1 close_window(GtkWidget *widget)	11
4.1.1.2 close_window_toggled()	11
4.1.1.3 createTask_pop_up(void)	12
4.1.1.4 device_window()	12
4.1.1.5 graph_button_clicked(GtkWidget *widget)	12
4.1.1.6 graph_clicked(GtkWidget *widget)	12
4.1.1.7 handle_task_menu(GtkWidget *widget, char *signal)	12
4.1.1.8 handle_task_prio(GtkWidget *widget, char *signal)	12
4.1.1.9 on_treeview_tasks_button_press_event(GtkButton *button, GdkEventButton *event)	12
4.1.1.10 process_window()	12
4.1.1.11 record_window()	12
4.1.1.12 show_all(GtkWidget *widget)	12
4.1.1.13 show_hide(GtkWidget *button, GtkWidget *window)	12
4.2 buttons.h File Reference	12
4.2.1 Function Documentation	14
4.2.1.1 close_window(GtkWidget *widget)	14
4.2.1.2 close_window_toggled()	14
4.2.1.3 close_window_v1(GtkWidget *widget)	14
4.2.1.4 createTask_pop_up(void)	14
4.2.1.5 device_window()	14
4.2.1.6 graph_button_clicked(GtkWidget *widget)	14
4.2.1.7 graph_clicked(GtkWidget *widget)	14
4.2.1.8 handle_task_menu(GtkWidget *widget, char *signal)	14
4.2.1.9 handle_task_prio(GtkWidget *widget, char *signal)	14
4.2.1.10 on_treeview_tasks_button_press_event(GtkButton *button, GdkEventButton *event)	14
4.2.1.11 process_window()	14

4.2.1.12	record_window()	14
4.2.1.13	show_all(GtkWidget *widget)	15
4.2.1.14	show_hide(GtkWidget *button, GtkWidget *window)	15
4.2.1.15	write_window()	15
4.2.2	Variable Documentation	15
4.2.2.1	button_dec	15
4.2.2.2	button_dev	15
4.2.2.3	button_device_all	15
4.2.2.4	button_device_avail	15
4.2.2.5	button_device_devices	15
4.2.2.6	button_device_directory	15
4.2.2.7	button_device_free	15
4.2.2.8	button_device_total	15
4.2.2.9	button_device_type	15
4.2.2.10	button_device_used	15
4.2.2.11	button_graph	15
4.2.2.12	button_inc	15
4.2.2.13	button_pause	15
4.2.2.14	button_proc	15
4.2.2.15	button_process_cpu	15
4.2.2.16	button_process_duration	15
4.2.2.17	button_process_pid	15
4.2.2.18	button_process_ppid	15
4.2.2.19	button_process_prio	15
4.2.2.20	button_process_rss	16
4.2.2.21	button_process_state	16
4.2.2.22	button_process_task	16
4.2.2.23	button_process_user	16
4.2.2.24	button_process_vm_size	16
4.2.2.25	button_rec	16

4.2.2.26	closed_cpu_window	16
4.2.2.27	cpu_buttons	16
4.2.2.28	dev_window	16
4.2.2.29	proc_window	16
4.2.2.30	rec_window	16
4.2.2.31	task_popup	16
4.2.2.32	wr_window	16
4.3	common.h File Reference	16
4.3.1	Macro Definition Documentation	17
4.3.1.1	CPU_PACK	17
4.3.1.2	CPU_USAGE	17
4.3.1.3	DEVICES	18
4.3.1.4	INT_PACK	18
4.3.1.5	INTERRUPTS	18
4.3.1.6	MEMORY	18
4.3.1.7	NETWORK	18
4.3.1.8	TASK	18
4.3.1.9	TEXT	18
4.3.2	Typedef Documentation	18
4.3.2.1	Cpu_usage	18
4.3.2.2	D_Collection	18
4.3.2.3	Data	18
4.3.2.4	Devices	18
4.3.2.5	I_Collection	18
4.3.2.6	I_Collection2	18
4.3.2.7	Interrupts	18
4.3.2.8	Interrupts2	18
4.3.2.9	Interrupts_send	18
4.3.2.10	Memory_usage	18
4.3.2.11	Network	18

4.3.2.12	T_Collection	18
4.3.2.13	Task	19
4.3.2.14	Unification	19
4.3.3	Function Documentation	19
4.3.3.1	__attribute__((__packed__)) tm1	19
4.3.4	Variable Documentation	19
4.3.4.1	fontdesc	19
4.4	drawing.c File Reference	19
4.4.1	Function Documentation	20
4.4.1.1	do_drawing_cpu(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array1)	20
4.4.1.2	do_drawing_int2(GtkWidget *widget, cairo_t *cr, I_Collection2 *interrupts1)	20
4.4.1.3	do_drawing_mem(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array)	20
4.4.1.4	do_drawing_net(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array)	20
4.4.1.5	do_drawing_one_cpu(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array1, int index)	20
4.4.1.6	draw_frame(cairo_t *cr, double width, double height, double font_size, int i)	20
4.4.1.7	draw_graph(cairo_t *cr, int r, double width, double height, double font_size, double time_step, Mega_Data *array)	20
4.4.1.8	draw_graph_mem(cairo_t *cr, int r, int i, double width, double height, double font_size, double time_step, Mega_Data *array)	20
4.4.1.9	draw_graph_net(cairo_t *cr, int r, int i, double width, double height, double font_size, double time_step, float max_num, Mega_Data *array)	20
4.4.1.10	draw_interrupts2(cairo_t *cr, int position, Interrupts2 *peak, double height, double font_size, __int64_t max_num, double length)	21
4.4.1.11	draw_percentages(cairo_t *cr, double height, double font_size)	21
4.4.1.12	on_draw_event(GtkWidget *widget, cairo_t *cr)	21
4.4.1.13	writing_interrupt_names2(cairo_t *cr, double font_size, double length, int position, const gchar *name1)	21
4.4.1.14	writing_seconds(cairo_t *cr, double width, double height, double font_size, int i)	21
4.5	drawing.h File Reference	21
4.5.1	Function Documentation	22

4.5.1.1	checking_interrupt_names(cairo_t *cr, double font_size, double length, int position, const char *ime1, const char *ime2, const char *name3, const char *name4)	22
4.5.1.2	do_drawing_cpu(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array1)	22
4.5.1.3	do_drawing_cpu2(GtkWidget *widget, cairo_t *cr, guint time_step, Cpu_list *array1)	22
4.5.1.4	do_drawing_int(GtkWidget *widget, cairo_t *cr, I_Collection *interrupts1)	22
4.5.1.5	do_drawing_int2(GtkWidget *widget, cairo_t *cr, I_Collection2 *interrupts1)	22
4.5.1.6	do_drawing_mem(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array)	22
4.5.1.7	do_drawing_net(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array)	22
4.5.1.8	do_drawing_one_cpu(GtkWidget *widget, cairo_t *cr, guint time_step, Mega_Data *array1, int index)	22
4.5.1.9	draw_frame(cairo_t *cr, double width, double height, double font_size, int i)	22
4.5.1.10	draw_graph(cairo_t *cr, int r, double width, double height, double font_size, double time_step, Mega_Data *array)	22
4.5.1.11	draw_graph_mem(cairo_t *cr, int r, int i, double width, double height, double font_size, double time_step, Mega_Data *array)	22
4.5.1.12	draw_graph_net(cairo_t *cr, int r, int i, double width, double height, double font_size, double time_step, float max_num, Mega_Data *array)	23
4.5.1.13	draw_interrupts(cairo_t *cr, int position, Interrupts *peak, double height, double font_size, __uint64_t max_num, double length)	23
4.5.1.14	draw_percentages(cairo_t *cr, double height, double font_size)	23
4.5.1.15	on_draw_event(GtkWidget *widget, cairo_t *cr)	23
4.5.1.16	on_draw_event2(GtkWidget *widget, cairo_t *cr, Cpu_list *array)	23
4.5.1.17	writing_interrupt_names(cairo_t *cr, double font_size, double length, int position, const gchar *name1, const gchar *name2)	23
4.5.1.18	writing_interrupt_names2(cairo_t *cr, double font_size, double length, int position, const gchar *name1)	23
4.5.1.19	writing_seconds(cairo_t *cr, double width, double height, double font_size, int i)	23
4.6	functions.c File Reference	23
4.6.1	Function Documentation	24
4.6.1.1	command_sender(char *text)	24
4.6.1.2	connection(char *argv1, char *argv2)	24

4.6.1.3	<code>data_transfer(int socket, Cpu_usage *cpu_usage, Network *network, Memory↵_usage *memory_usage, T_Collection **task_array, D_Collection **devices_↵array, __int32_t *task_num, __int32_t *dev_num, l_Collection2 **interrupts_p)</code>	24
4.6.1.4	<code>device_task_commands(char *signal, char *task_id)</code>	24
4.6.1.5	<code>input_command()</code>	24
4.6.1.6	<code>receive_number_cpu(int socket)</code>	24
4.6.1.7	<code>scan_numbers(__uint64_t *CPU, char *ptr, int *cpu_index)</code>	24
4.6.1.8	<code>test_rcv(int socket)</code>	24
4.6.1.9	<code>test_send(int socket)</code>	24
4.7	functions.h File Reference	25
4.7.1	Function Documentation	25
4.7.1.1	<code>command_sender(char *text)</code>	25
4.7.1.2	<code>connection(char *argv1, char *argv2)</code>	25
4.7.1.3	<code>data_transfer(int socket, Cpu_usage *cpu_usage, Network *network, Memory↵_usage *memory_usage, T_Collection **task_array, D_Collection **devices_↵array, __int32_t *task_num, __int32_t *dev_num, l_Collection2 **interrupts_p)</code>	25
4.7.1.4	<code>device_task_commands(char *signal, char *task_id)</code>	25
4.7.1.5	<code>input_command()</code>	26
4.7.1.6	<code>receive_number_cpu(int socket)</code>	26
4.7.1.7	<code>test_rcv(int socket)</code>	26
4.7.1.8	<code>test_send(int socket)</code>	26
4.8	main.c File Reference	26
4.8.1	Function Documentation	27
4.8.1.1	<code>dec_refresh()</code>	27
4.8.1.2	<code>destroy_window(void)</code>	27
4.8.1.3	<code>free_mega_data(Mega_Data **m_ptr)</code>	27
4.8.1.4	<code>free_one_mega_data(Mega_Data *m_ptr)</code>	27
4.8.1.5	<code>freeing_memory(void *array, __int32_t *array_size, int type)</code>	27
4.8.1.6	<code>inc_refresh()</code>	27
4.8.1.7	<code>init_timeout()</code>	27
4.8.1.8	<code>main(int argc, char *argv[])</code>	28
4.8.1.9	<code>pause_app(GtkWidget *button)</code>	28

4.8.1.10	set_record(GtkWidget *widget)	28
4.8.1.11	test_strtol(long val)	28
4.8.1.12	timeout_refresh()	28
4.8.2	Variable Documentation	28
4.8.2.1	app	28
4.8.2.2	flag_timeout	28
4.8.2.3	sem	28
4.8.2.4	window	28
4.8.2.5	writing	28
4.9	main_header.h File Reference	29
4.9.1	Macro Definition Documentation	30
4.9.1.1	LIST_SIZE	30
4.9.2	Typedef Documentation	30
4.9.2.1	Cpu_list	30
4.9.2.2	Mega_Data	30
4.9.2.3	NetMem_list	30
4.9.3	Function Documentation	30
4.9.3.1	dec_refresh()	30
4.9.3.2	destroy_window(void)	31
4.9.3.3	device_check(D_Collection *devices_new, int dev_num)	31
4.9.3.4	freeing_memory(void *array, __int32_t *array_size, int type)	31
4.9.3.5	inc_refresh()	31
4.9.3.6	init_timeout()	31
4.9.3.7	pause_app(GtkWidget *button)	31
4.9.3.8	set_record(GtkWidget *widget)	31
4.9.3.9	task_check(T_Collection *tasks_new, int task_num)	31
4.9.3.10	test_strtol(long val)	31
4.9.3.11	timeout_refresh()	31
4.9.4	Variable Documentation	31
4.9.4.1	cpu_list	31

4.9.4.2	cpu_num	32
4.9.4.3	cpu_status	32
4.9.4.4	dev_num_old	32
4.9.4.5	device_all	32
4.9.4.6	device_swindow	32
4.9.4.7	devices_old	32
4.9.4.8	entry	32
4.9.4.9	interrupt_num	32
4.9.4.10	interrupts	32
4.9.4.11	interrupts2	32
4.9.4.12	list_num_size	32
4.9.4.13	m_data	32
4.9.4.14	mem_list	33
4.9.4.15	net_list	33
4.9.4.16	newsockfd	33
4.9.4.17	newsockfd1	33
4.9.4.18	p_dir	33
4.9.4.19	process_swindow	33
4.9.4.20	record	33
4.9.4.21	refresh	33
4.9.4.22	t	33
4.9.4.23	task_num_old	33
4.9.4.24	tasks_old	33
4.9.4.25	time_step	33
4.9.4.26	window_graphs	33
4.10	testing.c File Reference	33
4.10.1	Function Documentation	34
4.10.1.1	cpu_read(Cpu_list **array)	34
4.10.1.2	cpu_write(Cpu_usage cpu_usage)	34
4.10.1.3	device_write(D_Collection *array)	34

4.10.1.4	<code>ifstat_calculate(float *received_kb, float *transmitted_kb)</code>	34
4.10.1.5	<code>interrupts_write(I_Collection2 *array)</code>	34
4.10.1.6	<code>memory_write(Memory_usage *memory_usage)</code>	34
4.10.1.7	<code>netw_calculate(float *transmitted, float *received)</code>	34
4.10.1.8	<code>netw_write(uint64_t transmitted, uint64_t received)</code>	34
4.10.1.9	<code>task_sort(T_Collection **array, int number)</code>	34
4.10.1.10	<code>task_write(T_Collection *array)</code>	34
4.11	testing.h File Reference	34
4.11.1	Function Documentation	35
4.11.1.1	<code>cpu_read(Cpu_list **array)</code>	35
4.11.1.2	<code>cpu_write(Cpu_usage cpu_usage)</code>	35
4.11.1.3	<code>device_write(D_Collection *array)</code>	35
4.11.1.4	<code>ifstat_calculate(float *received_kb, float *transmitted_kb)</code>	35
4.11.1.5	<code>interrupts_write(I_Collection2 *array)</code>	35
4.11.1.6	<code>memory_write(Memory_usage *memory_usage)</code>	35
4.11.1.7	<code>netw_calculate(float *transmitted, float *received)</code>	35
4.11.1.8	<code>netw_write(unsigned long transmitted, unsigned long received)</code>	35
4.11.1.9	<code>task_sort(T_Collection **array, int number)</code>	35
4.11.1.10	<code>task_write(T_Collection *array)</code>	35
4.12	testing_tree.c File Reference	35
4.12.1	Function Documentation	36
4.12.1.1	<code>add_new_dev(Devices *devices)</code>	36
4.12.1.2	<code>add_new_task(Task *task_t)</code>	36
4.12.1.3	<code>change_list_store_view_devices(GtkWidget *widget)</code>	36
4.12.1.4	<code>change_list_store_view_process(GtkWidget *widget)</code>	36
4.12.1.5	<code>compare_int_list_item(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	36
4.12.1.6	<code>compare_int_list_item_size(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	36
4.12.1.7	<code>compare_int_list_item_time(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	36

4.12.1.8	<code>compare_string_list_item(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	36
4.12.1.9	<code>create_list_store_dev(void)</code>	36
4.12.1.10	<code>create_list_store_task(void)</code>	36
4.12.1.11	<code>delete_old_dev(D_Collection **array, __int32_t *dev_num)</code>	36
4.12.1.12	<code>delete_old_tasks(T_Collection **array, __int32_t *task_num)</code>	36
4.12.1.13	<code>device_check(D_Collection *devices_new, int dev_num)</code>	36
4.12.1.14	<code>fill_device_item(Devices *f_temp, GtkTreeIter *iter)</code>	36
4.12.1.15	<code>fill_task_item(Task *task_item, GtkTreeIter *iter, int *array_i)</code>	36
4.12.1.16	<code>insert_new_devices(D_Collection **array, D_Collection *devices_new, __int32_t dev_num, __int32_t *old_number)</code>	36
4.12.1.17	<code>insert_new_tasks(T_Collection **array, T_Collection *tasks_new, __int32_t task_num, __int32_t *old_number)</code>	37
4.12.1.18	<code>refresh_devices_data(D_Collection *devices_new, __int32_t device_num)</code>	37
4.12.1.19	<code>refresh_list_item(Task *task_item, int *array_i)</code>	37
4.12.1.20	<code>refresh_list_item_device(Devices *ref_temp)</code>	37
4.12.1.21	<code>refresh_task_data(T_Collection *tasks_new, int task_num)</code>	37
4.12.1.22	<code>remove_list_item_device(gchar *directory, gchar *name)</code>	37
4.12.1.23	<code>remove_task_item(gint pid)</code>	37
4.12.1.24	<code>task_check(T_Collection *tasks_new, int task_num)</code>	37
4.13	<code>testing_tree.h</code> File Reference	37
4.13.1	Enumeration Type Documentation	38
4.13.1.1	anonymous enum	38
4.13.1.2	anonymous enum	38
4.13.2	Function Documentation	39
4.13.2.1	<code>add_new_dev(Devices *devices)</code>	39
4.13.2.2	<code>add_new_task(Task *task_t)</code>	39
4.13.2.3	<code>change_list_store_view_devices(GtkWidget *widget)</code>	39
4.13.2.4	<code>change_list_store_view_process(GtkWidget *widget)</code>	39
4.13.2.5	<code>compare_int_list_item(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	39
4.13.2.6	<code>compare_int_list_item_size(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	39

4.13.2.7	<code>compare_int_list_item_time(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	39
4.13.2.8	<code>compare_string_list_item(GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)</code>	39
4.13.2.9	<code>create_list_store_dev(void)</code>	39
4.13.2.10	<code>create_list_store_task(void)</code>	39
4.13.2.11	<code>delete_old_dev(D_Collection **array, __int32_t *dev_num)</code>	39
4.13.2.12	<code>delete_old_tasks(T_Collection **array, __int32_t *task_num)</code>	39
4.13.2.13	<code>device_check(D_Collection *devices_new, int dev_num)</code>	39
4.13.2.14	<code>fill_device_item(Devices *f_temp, GtkTreeIter *iter)</code>	39
4.13.2.15	<code>fill_task_item(Task *task_item, GtkTreeIter *iter, int *array_i)</code>	39
4.13.2.16	<code>refresh_list_item(Task *task_item, int *array_i)</code>	39
4.13.2.17	<code>refresh_list_item_device(Devices *ref_temp)</code>	39
4.13.2.18	<code>remove_list_item_device(gchar *directory, gchar *name)</code>	39
4.13.2.19	<code>remove_task_item(gint pid)</code>	39
4.13.2.20	<code>task_check(T_Collection *tasks_new, int task_num)</code>	39
4.13.3	Variable Documentation	39
4.13.3.1	<code>cell_renderer</code>	39
4.13.3.2	<code>list_devices</code>	39
4.13.3.3	<code>list_tasks</code>	39
4.13.3.4	<code>selection</code>	40
4.13.3.5	<code>treeview_devices</code>	40
4.13.3.6	<code>treeview_tasks</code>	40
4.14	window.c File Reference	40
4.14.1	Function Documentation	40
4.14.1.1	<code>cpu_change(Cpu_usage *cpu_usage)</code>	40
4.14.1.2	<code>cpu_window(int cpu_number, GtkWidget **graph_list)</code>	40
4.14.1.3	<code>main_window(GtkWidget *dev_swindow, GtkWidget *process_swindow)</code>	40
4.14.1.4	<code>memory_change(Memory_usage *memory_usage)</code>	40
4.14.1.5	<code>network_change(Network *network)</code>	40
4.14.1.6	<code>swap_change(Memory_usage *memory_usage)</code>	41

4.15 window.h File Reference	41
4.15.1 Function Documentation	41
4.15.1.1 cpu_change(Cpu_usage *cpu_usage)	41
4.15.1.2 cpu_window(int cpu_number, GtkWidget **graph_list)	41
4.15.1.3 main_window(GtkWidget *des_swindow, GtkWidget *proc_swindow)	41
4.15.1.4 memory_change(Memory_usage *memory_usage)	41
4.15.1.5 network_change(Network *network)	41
4.15.1.6 swap_change(Memory_usage *memory_usage)	42
4.15.2 Variable Documentation	42
4.15.2.1 adj	42
4.15.2.2 cpu_graphs	42
4.15.2.3 CPU_WINDOW	42
4.15.2.4 graph1	42
4.15.2.5 graph_intrp	42
4.15.2.6 graph_mem	42
4.15.2.7 graph_net	42
4.15.2.8 graph_write	42
4.15.2.9 interrupts_swindow	42
4.15.2.10 label_cpu0	42
4.15.2.11 label_mem	42
4.15.2.12 label_rec	42
4.15.2.13 label_swap	42
4.15.2.14 label_trans	42
4.15.2.15 viewport	42
Index	43

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Cpu_List	5
Device_Collection	5
Interrupt_Collection	6
Interrupt_Collection2	7
Mega_Data	7
NetMem_list	8
Task_Collection	9
Unification	9

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

buttons.c	11
buttons.h	12
common.h	16
drawing.c	19
drawing.h	21
functions.c	23
functions.h	25
main.c	26
main_header.h	29
testing.c	33
testing.h	34
testing_tree.c	35
testing_tree.h	37
window.c	40
window.h	41

Chapter 3

Data Structure Documentation

3.1 Cpu_List Struct Reference

```
#include <main_header.h>
```

Data Fields

- float * [data](#)
- [Cpu_list](#) * next

3.1.1 Detailed Description

structure for creating linked list for cpu usage

3.1.2 Field Documentation

3.1.2.1 float* data

3.1.2.2 Cpu_list* next

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

- [main_header.h](#)

3.2 Device_Collection Struct Reference

```
#include <common.h>
```

Data Fields

- [Devices devices](#)
- [D_Collection * next](#)
- [D_Collection * prev](#)

3.2.1 Detailed Description

doubly linked list for devices

3.2.2 Field Documentation

3.2.2.1 Devices devices

3.2.2.2 D_Collection* next

3.2.2.3 D_Collection* prev

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

- [common.h](#)

3.3 Interrupt_Collection Struct Reference

```
#include <common.h>
```

Data Fields

- [Interrupts interrupts](#)
- [I_Collection * next](#)
- [I_Collection * prev](#)

3.3.1 Field Documentation

3.3.1.1 Interrupts interrupts

3.3.1.2 I_Collection* next

3.3.1.3 I_Collection* prev

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

- [common.h](#)

3.4 Interrupt_Collection2 Struct Reference

```
#include <common.h>
```

Data Fields

- [Interrupts2 interrupts](#)
- [I_Collection2 * next](#)
- [I_Collection2 * prev](#)

3.4.1 Field Documentation

3.4.1.1 Interrupts2 interrupts

3.4.1.2 I_Collection2* next

3.4.1.3 I_Collection2* prev

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

- [common.h](#)

3.5 Mega_Data Struct Reference

```
#include <main_header.h>
```

Data Fields

- float * [cpu_stats](#)
- float * [mem_stats](#)
- float * [net_stats](#)
- [T_Collection](#) * [task_list](#)
- [D_Collection](#) * [device_list](#)
- [I_Collection2](#) * [interrupts_list](#)
- [Mega_Data](#) * [next](#)

3.5.1 Field Documentation

3.5.1.1 float* cpu_stats

3.5.1.2 D_Collection* device_list

3.5.1.3 I_Collection2* interrupts_list

3.5.1.4 float* mem_stats

3.5.1.5 float* net_stats

3.5.1.6 Mega_Data* next

3.5.1.7 T_Collection* task_list

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

- [main_header.h](#)

3.6 NetMem_list Struct Reference

```
#include <main_header.h>
```

Data Fields

- float [data](#) [2]
- [NetMem_list](#) * [next](#)

3.6.1 Detailed Description

structure for creating linked list for memory usage and network usage

3.6.2 Field Documentation

3.6.2.1 float data[2]

3.6.2.2 [NetMem_list](#)* next

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

- [main_header.h](#)

3.7 Task_Collection Struct Reference

```
#include <common.h>
```

Data Fields

- [Task task](#)
- [T_Collection * next](#)
- [T_Collection * prev](#)

3.7.1 Field Documentation

3.7.1.1 T_Collection* next

3.7.1.2 T_Collection* prev

3.7.1.3 Task task

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

- [common.h](#)

3.8 Unification Union Reference

```
#include <common.h>
```

Data Fields

- [Task task](#)
- [Network network](#)
- [Memory_usage memory_usage](#)
- [Interrupts interrupts](#)
- [Interrupts_send interrupts_send](#)
- [Devices devices](#)
- char [conformation](#) [64]
- char [data_pack](#) [1024]

3.8.1 Detailed Description

union data structure that uses the same memory space for all elements

3.8.2 Field Documentation

3.8.2.1 char conformation[64]

3.8.2.2 char data_pack[1024]

3.8.2.3 Devices devices

3.8.2.4 Interrupts interrupts

3.8.2.5 Interrupts_send interrupts_send

3.8.2.6 Memory_usage memory_usage

3.8.2.7 Network network

3.8.2.8 Task task

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

- [common.h](#)

Chapter 4

File Documentation

4.1 buttons.c File Reference

```
#include "buttons.h"
#include "drawing.h"
#include "testing_tree.h"
#include "functions.h"
```

Functions

- void [process_window](#) ()
- void [record_window](#) ()
- void [device_window](#) ()
- void [close_window_toggled](#) ()
- void [close_window](#) (GtkWidget *widget)
- void [graph_button_clicked](#) (GtkWidget *widget)
- void [show_all](#) (GtkWidget *widget)
- void [graph_clicked](#) (GtkWidget *widget)
- void [show_hide](#) (GtkWidget *button, GtkWidget *window)
- void [handle_task_menu](#) (GtkWidget *widget, char *signal)
- void [handle_task_prio](#) (GtkWidget *widget, char *signal)
- GtkWidget * [createTask_pop_up](#) (void)
- gboolean [on_treeview_tasks_button_press_event](#) (GtkButton *button, GdkEventButton *event)

4.1.1 Function Documentation

4.1.1.1 void [close_window](#) (GtkWidget * *widget*)

function [close_window\(\)](#): closes a widget input:pointer to a widget. output:none.

4.1.1.2 void [close_window_toggled](#) ()

function [close_window_toggled\(\)](#): when the graph window is closed we set the button graph to not be clicked
input:none. output:none.

4.1.1.3 `GtkWidget* createTask_pop_up (void)`

4.1.1.4 `void device_window ()`

function `device_window()`: create a window with buttons that represent columns in the device list when a button is checked the column connected to that button is shown or hidden. The buttons are checked depending on the columns visibility input:none. output:none.

4.1.1.5 `void graph_button_clicked (GtkWidget * widget)`

4.1.1.6 `void graph_clicked (GtkWidget * widget)`

4.1.1.7 `void handle_task_menu (GtkWidget * widget, char * signal)`

4.1.1.8 `void handle_task_prio (GtkWidget * widget, char * signal)`

4.1.1.9 `gboolean on_treeview_tasks_button_press_event (GtkWidget * button, GdkEventButton * event)`

function `on_treeview_tasks_button_press_event()`: reacts to a right click on a task in the list and then creates a pop up menu. input:gtk button and pointer to an event. output: return bool.

4.1.1.10 `void process_window ()`

function `process_window()`: create a window with buttons that represent columns in the task list when a button is checked the column connected to that button is shown or hidden. The buttons are checked depending on the columns visibility input:none. output:none.

4.1.1.11 `void record_window ()`

function `record_window()` creates a window that holds the record button input : none output : none

4.1.1.12 `void show_all (GtkWidget * widget)`

4.1.1.13 `void show_hide (GtkWidget * button, GtkWidget * window)`

4.2 buttons.h File Reference

```
#include <gtk/gtk.h>
#include <stdbool.h>
```

Functions

- gboolean [on_treeview_tasks_button_press_event](#) (GtkButton *button, GdkEventButton *event)
- GtkWidget * [createTask_pop_up](#) (void)
- void [handle_task_menu](#) (GtkWidget *widget, char *signal)
- void [handle_task_prio](#) (GtkWidget *widget, char *signal)
- void [graph_button_clicked](#) (GtkWidget *widget)
- void [close_window_toggled](#) ()
- void [close_window](#) (GtkWidget *widget)
- void [close_window_v1](#) (GtkWidget *widget)
- void [show_hide](#) (GtkWidget *button, GtkWidget *[window](#))
- void [device_window](#) ()
- void [process_window](#) ()
- void [record_window](#) ()
- void [graph_clicked](#) (GtkWidget *widget)
- void [show_all](#) (GtkWidget *widget)
- void [write_window](#) ()

Variables

- GtkWidget * [dev_window](#)
- GtkWidget * [proc_window](#)
- GtkWidget * [rec_window](#)
- GtkWidget * [wr_window](#)
- bool [closed_cpu_window](#)
- GtkWidget * [task_popup](#)
- GtkWidget * [button_inc](#)
- GtkWidget * [button_dec](#)
- GtkWidget * [button_proc](#)
- GtkWidget * [button_dev](#)
- GtkWidget * [button_rec](#)
- GtkWidget * [button_graph](#)
- GtkWidget * [button_pause](#)
- GtkWidget * [button_device_devices](#)
- GtkWidget * [button_device_directory](#)
- GtkWidget * [button_device_type](#)
- GtkWidget * [button_device_avail](#)
- GtkWidget * [button_device_used](#)
- GtkWidget * [button_device_all](#)
- GtkWidget * [button_device_free](#)
- GtkWidget * [button_device_total](#)
- GtkWidget * [button_process_task](#)
- GtkWidget * [button_process_user](#)
- GtkWidget * [button_process_pid](#)
- GtkWidget * [button_process_ppid](#)
- GtkWidget * [button_process_state](#)
- GtkWidget * [button_process_vm_size](#)
- GtkWidget * [button_process_rss](#)
- GtkWidget * [button_process_cpu](#)
- GtkWidget * [button_process_prio](#)
- GtkWidget * [button_process_duration](#)
- GtkWidget * [cpu_buttons](#)

4.2.1 Function Documentation

4.2.1.1 void close_window (GtkWidget * widget)

function [close_window\(\)](#): closes a widget input:pointer to a widget. output:none.

4.2.1.2 void close_window_toggled ()

function [close_window_toggled\(\)](#): when the graph window is closed we set the button graph to not be clicked input:none. output:none.

4.2.1.3 void close_window_v1 (GtkWidget * widget)

4.2.1.4 GtkWidget* createTask_pop_up (void)

4.2.1.5 void device_window ()

function [device_window\(\)](#): create a window with buttons that represent columns in the device list when a button is checked the column connected to that button is shown or hidden. The buttons are checked depending on the columns visibility input:none. output:none.

4.2.1.6 void graph_button_clicked (GtkWidget * widget)

4.2.1.7 void graph_clicked (GtkWidget * widget)

4.2.1.8 void handle_task_menu (GtkWidget * widget, char * signal)

4.2.1.9 void handle_task_prio (GtkWidget * widget, char * signal)

4.2.1.10 gboolean on_treeview_tasks_button_press_event (GtkWidget * button, GdkEventButton * event)

function [on_treeview_tasks_button_press_event\(\)](#): reacts to a right click on a task in the list and then creates a pop up menu. input:gtk button and pointer to an event. output: return bool.

4.2.1.11 void process_window ()

function [process_window\(\)](#): create a window with buttons that represent columns in the task list when a button is checked the column connected to that button is shown or hidden. The buttons are checked depending on the columns visibility input:none. output:none.

4.2.1.12 void record_window ()

function [record_window\(\)](#) creates a window that holds the record button input : none output : none

4.2.1.13 void show_all (GtkWidget * *widget*)

4.2.1.14 void show_hide (GtkWidget * *button*, GtkWidget * *window*)

4.2.1.15 void write_window ()

4.2.2 Variable Documentation

4.2.2.1 GtkWidget* button_dec

4.2.2.2 GtkWidget* button_dev

4.2.2.3 GtkWidget* button_device_all

4.2.2.4 GtkWidget* button_device_avail

4.2.2.5 GtkWidget* button_device_devices

4.2.2.6 GtkWidget* button_device_directory

4.2.2.7 GtkWidget* button_device_free

4.2.2.8 GtkWidget* button_device_total

4.2.2.9 GtkWidget* button_device_type

4.2.2.10 GtkWidget* button_device_used

4.2.2.11 GtkWidget* button_graph

4.2.2.12 GtkWidget* button_inc

4.2.2.13 GtkWidget* button_pause

4.2.2.14 GtkWidget* button_proc

4.2.2.15 GtkWidget* button_process_cpu

4.2.2.16 GtkWidget* button_process_duration

4.2.2.17 GtkWidget* button_process_pid

4.2.2.18 GtkWidget* button_process_ppid

4.2.2.19 GtkWidget* button_process_prio

4.2.2.20 `GtkWidget*` `button_process_rss`

4.2.2.21 `GtkWidget*` `button_process_state`

4.2.2.22 `GtkWidget*` `button_process_task`

4.2.2.23 `GtkWidget*` `button_process_user`

4.2.2.24 `GtkWidget*` `button_process_vm_size`

4.2.2.25 `GtkWidget*` `button_rec`

4.2.2.26 `bool` `closed_cpu_window`

4.2.2.27 `GtkWidget*` `cpu_buttons`

4.2.2.28 `GtkWidget*` `dev_window`

4.2.2.29 `GtkWidget*` `proc_window`

4.2.2.30 `GtkWidget*` `rec_window`

4.2.2.31 `GtkWidget*` `task_popup`

4.2.2.32 `GtkWidget*` `wr_window`

4.3 common.h File Reference

```
#include <stdbool.h>
#include <time.h>
#include <cairo.h>
#include <gtk/gtk.h>
```

Data Structures

- struct [Device_Collection](#)
- struct [Task_Collection](#)
- struct [Interrupt_Collection](#)
- struct [Interrupt_Collection2](#)
- union [Unification](#)

Macros

- `#define CPU_USAGE 1`
- `#define NETWORK 2`
- `#define MEMORY 3`
- `#define TASK 4`
- `#define DEVICES 5`
- `#define INTERRUPTS 6`
- `#define TEXT 7`
- `#define CPU_PACK 8`
- `#define INT_PACK 9`

Typedefs

- `typedef struct Task Task`
- `typedef struct Network Network`
- `typedef struct Cpu_usage Cpu_usage`
- `typedef struct Memory_usage Memory_usage`
- `typedef struct Interrupts_send Interrupts_send`
- `typedef struct Interrupts2 Interrupts2`
- `typedef struct Interrupts Interrupts`
- `typedef struct Devices Devices`
- `typedef struct Device_Collection D_Collection`
- `typedef struct Task_Collection T_Collection`
- `typedef struct Interrupt_Collection I_Collection`
- `typedef struct Interrupt_Collection2 I_Collection2`
- `typedef union Unification Unification`
- `typedef struct Data Data`

Functions

- `struct __attribute__((__packed__)) tm1`

Variables

- `PangoFontDescription * fontdesc`

4.3.1 Macro Definition Documentation

4.3.1.1 `#define CPU_PACK 8`

4.3.1.2 `#define CPU_USAGE 1`

defines what type of data we are sending

4.3.1.3 `#define DEVICES 5`

4.3.1.4 `#define INT_PACK 9`

4.3.1.5 `#define INTERRUPTS 6`

4.3.1.6 `#define MEMORY 3`

4.3.1.7 `#define NETWORK 2`

4.3.1.8 `#define TASK 4`

4.3.1.9 `#define TEXT 7`

4.3.2 Typedef Documentation

4.3.2.1 `typedef struct Cpu_usage Cpu_usage`

4.3.2.2 `typedef struct Device_Collection D_Collection`

4.3.2.3 `typedef struct Data Data`

4.3.2.4 `typedef struct Devices Devices`

4.3.2.5 `typedef struct Interrupt_Collection I_Collection`

4.3.2.6 `typedef struct Interrupt_Collection2 I_Collection2`

4.3.2.7 `typedef struct Interrupts Interrupts`

structure that contains the information of a interrupt type

4.3.2.8 `typedef struct Interrupts2 Interrupts2`

4.3.2.9 `typedef struct Interrupts_send Interrupts_send`

4.3.2.10 `typedef struct Memory_usage Memory_usage`

4.3.2.11 `typedef struct Network Network`

4.3.2.12 `typedef struct Task_Collection T_Collection`

doubly linked list for tasks

4.3.2.13 typedef struct Task Task

<structure that contains task details

4.3.2.14 typedef union Unification Unification

4.3.3 Function Documentation

4.3.3.1 struct __attribute__((__packed__))

structure that contains all the network usage

structure that contains information about memory usage

the structure we use to send data structure that contains cpu usage of all the different cpus

4.3.4 Variable Documentation

4.3.4.1 PangoFontDescription* fontdesc

4.4 drawing.c File Reference

```
#include <inttypes.h>
#include <errno.h>
#include "drawing.h"
#include "buttons.h"
#include <assert.h>
```

Functions

- gboolean [on_draw_event](#) (GtkWidget *widget, cairo_t *cr)
- void [writing_interrupt_names2](#) (cairo_t *cr, double font_size, double length, int position, const gchar *name1)
- void [writing_seconds](#) (cairo_t *cr, double width, double height, double font_size, int i)
- void [draw_frame](#) (cairo_t *cr, double width, double height, double font_size, int i)
- void [draw_percentages](#) (cairo_t *cr, double height, double font_size)
- void [draw_interrupts2](#) (cairo_t *cr, int position, [Interrupts2](#) *peak, double height, double font_size, __int64_t max_num, double length)
- void [draw_graph](#) (cairo_t *cr, int r, double width, double height, double font_size, double [time_step](#), [Mega_Data](#) *array)
- void [draw_graph_net](#) (cairo_t *cr, int r, int i, double width, double height, double font_size, double [time_step](#), float max_num, [Mega_Data](#) *array)
- void [draw_graph_mem](#) (cairo_t *cr, int r, int i, double width, double height, double font_size, double [time_step](#), [Mega_Data](#) *array)
- void [do_drawing_mem](#) (GtkWidget *widget, cairo_t *cr, guint [time_step](#), [Mega_Data](#) *array)
- void [do_drawing_int2](#) (GtkWidget *widget, cairo_t *cr, [I_Collection2](#) *interrupts1)
- void [do_drawing_net](#) (GtkWidget *widget, cairo_t *cr, guint [time_step](#), [Mega_Data](#) *array)
- void [do_drawing_one_cpu](#) (GtkWidget *widget, cairo_t *cr, guint [time_step](#), [Mega_Data](#) *array1, int index)
- void [do_drawing_cpu](#) (GtkWidget *widget, cairo_t *cr, guint [time_step](#), [Mega_Data](#) *array1)

4.4.1 Function Documentation

4.4.1.1 void do_drawing_cpu (GtkWidget * *widget*, cairo_t * *cr*, guint *time_step*, Mega_Data * *array1*)

function [do_drawing_cpu\(\)](#): draws the entire graph, the lines, the frame the seconds and the percentage input: pointer to the graph, pointer to the canvas, step between data, and pointer to the array of cpu usage output: none. display the cpus we want to be displayed

4.4.1.2 void do_drawing_int2 (GtkWidget * *widget*, cairo_t * *cr*, I_Collection2 * *interrupts1*)

4.4.1.3 void do_drawing_mem (GtkWidget * *widget*, cairo_t * *cr*, guint *time_step*, Mega_Data * *array*)

4.4.1.4 void do_drawing_net (GtkWidget * *widget*, cairo_t * *cr*, guint *time_step*, Mega_Data * *array*)

function [do_drawing_net\(\)](#): draws the entire graph, by searching the array for the biggest number input: pointer to the graph, pointer to the canvas, step between data, and pointer to the array of network usage output: none. how many elements do we have in an array

searching for the highest number in network usage

4.4.1.5 void do_drawing_one_cpu (GtkWidget * *widget*, cairo_t * *cr*, guint *time_step*, Mega_Data * *array1*, int *index*)

4.4.1.6 void draw_frame (cairo_t * *cr*, double *width*, double *height*, double *font_size*, int *i*)

function [draw_frame\(\)](#): drawing the frame of the graph input: pointer to the canvas, width of the graph, height of the graph, font size and position output: none.

4.4.1.7 void draw_graph (cairo_t * *cr*, int *r*, double *width*, double *height*, double *font_size*, double *time_step*, Mega_Data * *array*)

function [draw_graph\(\)](#): draws the lines on the graph input: pointer to the canvas, index of the data, width, height, font size, step between data, pointer to the array of data output: none.

4.4.1.8 void draw_graph_mem (cairo_t * *cr*, int *r*, int *i*, double *width*, double *height*, double *font_size*, double *time_step*, Mega_Data * *array*)

function [draw_graph_mem\(\)](#): draws the lines on the graph for memory usage input: pointer to the canvas, index of the data, type of graph, width, height, font size, step between data, max number for (network usage), pointer to the array of data output: none.

4.4.1.9 void draw_graph_net (cairo_t * *cr*, int *r*, int *i*, double *width*, double *height*, double *font_size*, double *time_step*, float *max_num*, Mega_Data * *array*)

function [draw_graph_net\(\)](#): draws the lines on the graph for network usage input: pointer to the canvas, index of the data, type of graph, width, height, font size, step between data, max number for (network usage), pointer to the array of data output: none. < the last line always touches the floor

4.4.1.10 void `draw_interrupts2` (`cairo_t * cr`, int *position*, `Interrupts2 * peak`, double *height*, double *font_size*, `__int64_t max_num`, double *length*)

4.4.1.11 void `draw_percentages` (`cairo_t * cr`, double *height*, double *font_size*)

4.4.1.12 gboolean `on_draw_event` (`GtkWidget * widget`, `cairo_t * cr`)

function `on_draw_event()`: creating graphs by sending the draw signal to the function we create a `cairo_t` structure
input: pointer to the graph ,and pointer to `cairo_t` on which to draw on output: if successful return true

4.4.1.13 void `writing_interrupt_names2` (`cairo_t * cr`, double *font_size*, double *length*, int *position*, const gchar * *name1*)

4.4.1.14 void `writing_seconds` (`cairo_t * cr`, double *width*, double *height*, double *font_size*, int *i*)

4.5 drawing.h File Reference

```
#include <math.h>
#include <sys/stat.h>
#include <pwd.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <gtk/gtk.h>
#include <cairo.h>
#include "common.h"
#include "main_header.h"
#include "window.h"
```

Functions

- gboolean `on_draw_event2` (`GtkWidget *widget`, `cairo_t *cr`, `Cpu_list *array`)
- gboolean `on_draw_event` (`GtkWidget *widget`, `cairo_t *cr`)
- void `do_drawing_net` (`GtkWidget *widget`, `cairo_t *cr`, guint *time_step*, `Mega_Data *array`)
- void `draw_graph_net` (`cairo_t *cr`, int *r*, int *i*, double *width*, double *height*, double *font_size*, double *time_step*, float *max_num*, `Mega_Data *array`)
- void `do_drawing_cpu` (`GtkWidget *widget`, `cairo_t *cr`, guint *time_step*, `Mega_Data *array1`)
- void `do_drawing_one_cpu` (`GtkWidget *widget`, `cairo_t *cr`, guint *time_step*, `Mega_Data *array1`, int *index*)
- void `do_drawing_mem` (`GtkWidget *widget`, `cairo_t *cr`, guint *time_step*, `Mega_Data *array`)
- void `draw_graph_mem` (`cairo_t *cr`, int *r*, int *i*, double *width*, double *height*, double *font_size*, double *time_step*, `Mega_Data *array`)
- void `do_drawing_int` (`GtkWidget *widget`, `cairo_t *cr`, `I_Collection *interrupts1`)
- void `do_drawing_int2` (`GtkWidget *widget`, `cairo_t *cr`, `I_Collection2 *interrupts1`)
- void `writing_seconds` (`cairo_t *cr`, double *width*, double *height*, double *font_size*, int *i*)
- void `draw_frame` (`cairo_t *cr`, double *width*, double *height*, double *font_size*, int *i*)
- void `draw_percentages` (`cairo_t *cr`, double *height*, double *font_size*)
- void `draw_interrupts` (`cairo_t *cr`, int *position*, `Interrupts *peak`, double *height*, double *font_size*, `__uint64_t max_num`, double *length*)
- void `writing_interrupt_names` (`cairo_t *cr`, double *font_size*, double *length*, int *position*, const gchar **name1*, const gchar **name2*)
- void `writing_interrupt_names2` (`cairo_t *cr`, double *font_size*, double *length*, int *position*, const gchar **name1*)
- void `checking_interrupt_names` (`cairo_t *cr`, double *font_size*, double *length*, int *position*, const char **ime1*, const char **ime2*, const char **name3*, const char **name4*)
- void `draw_graph` (`cairo_t *cr`, int *r*, double *width*, double *height*, double *font_size*, double *time_step*, `Mega_Data *array`)
- void `do_drawing_cpu2` (`GtkWidget *widget`, `cairo_t *cr`, guint *time_step*, `Cpu_list *array1`)

4.5.1 Function Documentation

4.5.1.1 void `checking_interrupt_names` (cairo_t * *cr*, double *font_size*, double *length*, int *position*, const char * *ime1*, const char * *ime2*, const char * *name3*, const char * *name4*)

4.5.1.2 void `do_drawing_cpu` (GtkWidget * *widget*, cairo_t * *cr*, quint *time_step*, Mega_Data * *array1*)

function `do_drawing_cpu()`: draws the entire graph, the lines, the frame the seconds and the percentage input: pointer to the graph, pointer to the canvas, step between data, and pointer to the array of cpu usage output: none. display the cpus we want to be displayed

4.5.1.3 void `do_drawing_cpu2` (GtkWidget * *widget*, cairo_t * *cr*, quint *time_step*, Cpu_list * *array1*)

4.5.1.4 void `do_drawing_int` (GtkWidget * *widget*, cairo_t * *cr*, I_Collection * *interrupts1*)

4.5.1.5 void `do_drawing_int2` (GtkWidget * *widget*, cairo_t * *cr*, I_Collection2 * *interrupts1*)

4.5.1.6 void `do_drawing_mem` (GtkWidget * *widget*, cairo_t * *cr*, quint *time_step*, Mega_Data * *array*)

4.5.1.7 void `do_drawing_net` (GtkWidget * *widget*, cairo_t * *cr*, quint *time_step*, Mega_Data * *array*)

function `do_drawing_net()`: draws the entire graph, by searching the array for the biggest number input: pointer to the graph, pointer to the canvas, step between data, and pointer to the array of network usage output: none. how many elements do we have in an array

searching for the highest number in network usage

4.5.1.8 void `do_drawing_one_cpu` (GtkWidget * *widget*, cairo_t * *cr*, quint *time_step*, Mega_Data * *array1*, int *index*)

4.5.1.9 void `draw_frame` (cairo_t * *cr*, double *width*, double *height*, double *font_size*, int *i*)

function `draw_frame()`: drawing the frame of the graph input: pointer to the canvas, width of the graph, height of the graph, font size and position output: none.

4.5.1.10 void `draw_graph` (cairo_t * *cr*, int *r*, double *width*, double *height*, double *font_size*, double *time_step*, Mega_Data * *array*)

function `draw_graph()`: draws the lines on the graph input: pointer to the canvas, index of the data, width, height, font size, step between data, pointer to the array of data output: none.

4.5.1.11 void `draw_graph_mem` (cairo_t * *cr*, int *r*, int *i*, double *width*, double *height*, double *font_size*, double *time_step*, Mega_Data * *array*)

function `draw_graph_mem()`: draws the lines on the graph for memory usage input: pointer to the canvas, index of the data, type of graph, width, height, font size, step between data, max number for (network usage), pointer to the array of data output: none.

4.5.1.12 `void draw_graph_net (cairo_t * cr, int r, int i, double width, double height, double font_size, double time_step, float max_num, Mega_Data * array)`

function `draw_graph_net()`: draws the lines on the graph for network usage input: pointer to the canvas, index of the data, type of graph, width, height, font size, step between data, max number for (network usage), pointer to the array of data output: none. < the last line always touches the floor

4.5.1.13 `void draw_interrupts (cairo_t * cr, int position, Interrupts * peak, double height, double font_size, __uint64_t max_num, double length)`

4.5.1.14 `void draw_percentages (cairo_t * cr, double height, double font_size)`

4.5.1.15 `gboolean on_draw_event (GtkWidget * widget, cairo_t * cr)`

function `on_draw_event()`: creating graphs by sending the draw signal to the function we create a `cairo_t` structure input: pointer to the graph , and pointer to `cairo_t` on which to draw on output: if successful return true

4.5.1.16 `gboolean on_draw_event2 (GtkWidget * widget, cairo_t * cr, Cpu_list * array)`

4.5.1.17 `void writing_interrupt_names (cairo_t * cr, double font_size, double length, int position, const gchar * name1, const gchar * name2)`

4.5.1.18 `void writing_interrupt_names2 (cairo_t * cr, double font_size, double length, int position, const gchar * name1)`

4.5.1.19 `void writing_seconds (cairo_t * cr, double width, double height, double font_size, int i)`

4.6 functions.c File Reference

```
#include "functions.h"
#include <memory.h>
#include <stdlib.h>
#include <errno.h>
#include <netdb.h>
#include "sys/socket.h"
#include "main_header.h"
```

Functions

- bool `scan_numbers` (__uint64_t *CPU, char *ptr, int *cpu_index)
- void `device_task_commands` (char *signal, char *task_id)
- int `connection` (char *argv1, char *argv2)
- void `input_command` ()
- int `command_sender` (char *text)
- ssize_t `test_send` (int socket)
- ssize_t `test_recv` (int socket)
- int `data_transfer` (int socket, Cpu_usage *cpu_usage, Network *network, Memory_usage *memory_usage, T_Collection **task_array, D_Collection **devices_array, __int32_t *task_num, __int32_t *dev_num, I_Collection2 **interrupts_p)
- long `receive_number_cpu` (int socket)

4.6.1 Function Documentation

4.6.1.1 `int command_sender (char * text)`

function `command_sender()`: prepares a text command to be sent and sends it to server input: none output: return non zero value if something is wrong

4.6.1.2 `int connection (char * argv1, char * argv2)`

function `connection()`: establishes a connection with the server input: port number and IP address output: return non zero value if something is wrong

4.6.1.3 `int data_transfer (int socket, Cpu_usage * cpu_usage, Network * network, Memory_usage * memory_usage, T_Collection ** task_array, D_Collection ** devices_array, __int32_t * task_num, __int32_t * dev_num, I_Collection2 ** interrupts_p)`

function `data_transfer()`: receives TCP packets from the server and handles them depending on the type of file they are input: socket, pointer cpu usage structure, pointer to network usage structure, double pointer to a Task doubly linked list, double pointer to a Device doubly linked list, tasks number and devices number output: return non zero value if something is wrong

4.6.1.4 `void device_task_commands (char * signal, char * task_id)`

function `device_task_commands()`: sends command to server about what type of devices it wants to see input: pointer to signal and to task id output: none.

4.6.1.5 `void input_command ()`

function `input_command()`: takes what we have typed in the entry widget and sends it to the server input: none output: none

4.6.1.6 `long receive_number_cpu (int socket)`

4.6.1.7 `bool scan_numbers (__uint64_t * CPU, char * ptr, int * cpu_index)`

4.6.1.8 `ssize_t test_recv (int socket)`

function `test_send()`: tests if the client can send TCP packets input: socket output: return non zero value if something is wrong

4.6.1.9 `ssize_t test_send (int socket)`

function `test_send()`: tests if the server can send TCP packets input: socket output: return non zero value if something is wrong

4.7 functions.h File Reference

```
#include "gtk/gtk.h"
#include "common.h"
```

Functions

- ssize_t [test_rcv](#) (int socket)
- ssize_t [test_send](#) (int socket)
- void [device_task_commands](#) (char *signal, char *task_id)
- int [data_transfer](#) (int socket, [Cpu_usage](#) *cpu_usage, [Network](#) *network, [Memory_usage](#) *memory_usage, [T_Collection](#) **task_array, [D_Collection](#) **devices_array, __int32_t *task_num, __int32_t *dev_num, [I_Collection2](#) **interrupts_p)
- int [command_sender](#) (char *text)
- void [input_command](#) ()
- int [connection](#) (char *argv1, char *argv2)
- long [receive_number_cpu](#) (int socket)

4.7.1 Function Documentation

4.7.1.1 int [command_sender](#) (char * *text*)

function [command_sender\(\)](#): prepares a text command to be sent and sends it to server input: none output: return non zero value if something is wrong

4.7.1.2 int [connection](#) (char * *argv1*, char * *argv2*)

function [connection\(\)](#): establishes a connection with the server input: port number and IP address output: return non zero value if something is wrong

4.7.1.3 int [data_transfer](#) (int *socket*, [Cpu_usage](#) * *cpu_usage*, [Network](#) * *network*, [Memory_usage](#) * *memory_usage*, [T_Collection](#) ** *task_array*, [D_Collection](#) ** *devices_array*, __int32_t * *task_num*, __int32_t * *dev_num*, [I_Collection2](#) ** *interrupts_p*)

function [data_transfer\(\)](#): receives TCP packets from the server and handles them depending on the type of file they are input: socket, pointer cpu usage structure, pointer to network usage structure, double pointer to a Task doubly linked list, double pointer to a Device doubly linked list, tasks number and devices number output: return non zero value if something is wrong

4.7.1.4 void [device_task_commands](#) (char * *signal*, char * *task_id*)

function [device_task_commands\(\)](#): sends command to server about what type of devices it wants to see input: pointer to signal and to task id output: none.

4.7.1.5 void input_command ()

function [input_command\(\)](#): takes what we have typed in the entry widget and sends it to the server input: none output:none

4.7.1.6 long receive_number_cpu (int socket)

4.7.1.7 ssize_t test_rcv (int socket)

function [test_send\(\)](#): tests if the client can send TCP packets input: socket output:return non zero value if something is wrong

4.7.1.8 ssize_t test_send (int socket)

function [test_send\(\)](#): tests if the server can send TCP packets input: socket output:return non zero value if something is wrong

4.8 main.c File Reference

```
#include "drawing.h"
#include "testing_tree.h"
#include "buttons.h"
#include <errno.h>
#include "functions.h"
#include "testing.h"
#include <semaphore.h>
#include <asm/errno.h>
#include <inttypes.h>
#include <fontconfig/fontconfig.h>
```

Functions

- void [set_record](#) (GtkWidget *widget)
- void [inc_refresh](#) ()
- void [dec_refresh](#) ()
- void [pause_app](#) (GtkWidget *button)
- void [timeout_refresh](#) ()
- void [freeing_memory](#) (void *array, __int32_t *array_size, int type)
- void [free_one_mega_data](#) (Mega_Data *m_ptr)
- void [free_mega_data](#) (Mega_Data **m_ptr)
- gboolean [init_timeout](#) ()
- void [destroy_window](#) (void)
- void [test_strtol](#) (long val)
- int [main](#) (int argc, char *argv[])

Variables

- GtkWidget * [window](#)
main window
- GtkApplication * [app](#)
application
- sem_t [semt](#)
semaphore for letting the `init_timeout` function finish before we change the time interval
- bool [flag_timeout](#) =true
flag for letting the `init_timeout` function know what to do
- bool [writing](#) =true
is recording being done

4.8.1 Function Documentation

4.8.1.1 void `dec_refresh` ()

function `dec_refresh()`: decrease the time that we want the client to request data from server input : none. output : none.

4.8.1.2 void `destroy_window` (void)

4.8.1.3 void `free_mega_data` (Mega_Data ** *m_ptr*)

4.8.1.4 void `free_one_mega_data` (Mega_Data * *m_ptr*)

4.8.1.5 void `freeing_memory` (void * *array*, __int32_t * *array_size*, int *type*)

function `freeing_memory()`: frees different types of memory input : void pointer to an array, pointer to the size of the array and the type of the array. output : none.

4.8.1.6 void `inc_refresh` ()

function `inc_refresh()`: increments the time that we want the client to request data from server input : none. output : none.

4.8.1.7 gboolean `init_timeout` ()

function `init_timeout()`: sends a request to server and then waits for data,after it got all the data it inputs it in the right places and checks if the `list_num_size` is bigger then the `LIST_SIZE` if that is the case it removes the oldest element of the list and adds the newest to the begging.After the data has been properly handled it displays it in the lists and draws the new data on the graph.We check if the function is running in an infinite loop,if not we set it to run in regular intervals that we have set. input : none output : returns TRUE if we want to continue or FALSE if we want to stop;

4.8.1.8 `int main (int argc, char * argv[])`

function `main()`: creates a TPC socket and tries to connect to the server,if that was successful it initializes the window and starts to request for data from the server;

input : port number and IP address output : returns a non zero value if something goes wrong

4.8.1.9 `void pause_app (GtkWidget * button)`

4.8.1.10 `void set_record (GtkWidget * widget)`

function `set_record()`: sets the record flag to true or false depending on if the button is clicked or not input : widget. output : none.

4.8.1.11 `void test_strtol (long val)`

4.8.1.12 `void timeout_refresh ()`

function `timeout_refresh()`: reruns the function `init_timeout` and tells the previous version to stop input : none. output : none.

4.8.2 Variable Documentation

4.8.2.1 `GtkApplication* app`

application

4.8.2.2 `bool flag_timeout =true`

flag for letting the `init_timeout` function know what to do

4.8.2.3 `sem_t semt`

semaphore for letting the `init_timeout` function finish before we change the time interval

4.8.2.4 `GtkWidget* window`

main window

4.8.2.5 `bool writing =true`

is recording being done

4.9 main_header.h File Reference

```
#include <gtk/gtk.h>
#include "common.h"
```

Data Structures

- struct [NetMem_list](#)
- struct [Cpu_List](#)
- struct [Mega_Data](#)

Macros

- `#define LIST_SIZE 240 /*!the max size of list of cpu, network and memory usage*/`

Typedefs

- typedef struct [NetMem_list](#) [NetMem_list](#)
main header please work
- typedef struct [Cpu_List](#) [Cpu_list](#)
- typedef struct [Mega_Data](#) [Mega_Data](#)

Functions

- gboolean [init_timeout](#) ()
- void [dec_refresh](#) ()
- void [inc_refresh](#) ()
- void [timeout_refresh](#) ()
- int [device_check](#) ([D_Collection](#) *devices_new, int dev_num)
- int [task_check](#) ([T_Collection](#) *tasks_new, int task_num)
- void [destroy_window](#) (void)
- void [freeing_memory](#) (void *array, __int32_t *array_size, int type)
- void [test_strtol](#) (long val)
- void [set_record](#) (GtkWidget *widget)
- void [pause_app](#) (GtkWidget *button)

Variables

- GtkWidget * [window_graphs](#)
- GtkWidget * [process_swindow](#)
- GtkWidget * [device_swindow](#)
- GtkWidget * [entry](#)
- int [newsockfd](#)
- int [newsockfd1](#)
- guint [t](#)
- guint [refresh](#)
- guint [time_step](#)
- __int32_t [dev_num_old](#)
- __int32_t [task_num_old](#)
- __int32_t [list_num_size](#)
- bool * [cpu_status](#)
- bool [device_all](#)
- bool [record](#)
- D_Collection * [devices_old](#)
- T_Collection * [tasks_old](#)
- I_Collection * [interrupts](#)
- I_Collection2 * [interrupts2](#)
- Cpu_list * [cpu_list](#)
- NetMem_list * [net_list](#)
- NetMem_list * [mem_list](#)
- Mega_Data * [m_data](#)
- char [p_dir](#) [256]
- long [cpu_num](#)
- long [interrupt_num](#)

4.9.1 Macro Definition Documentation

4.9.1.1 `#define LIST_SIZE 240 /*!the max size of list of cpu, network and memory usage*/`

4.9.2 Typedef Documentation

4.9.2.1 `typedef struct Cpu_List Cpu_list`

4.9.2.2 `typedef struct Mega_Data Mega_Data`

4.9.2.3 `typedef struct NetMem_list NetMem_list`

main header please work

4.9.3 Function Documentation

4.9.3.1 `void dec_refresh ()`

function [dec_refresh\(\)](#): decrease the time that we want the client to request data from server input : none. output : none.

4.9.3.2 void destroy_window (void)

4.9.3.3 int device_check (D_Collection * *devices_new*, int *dev_num*)

4.9.3.4 void freeing_memory (void * *array*, __int32_t * *array_size*, int *type*)

function [freeing_memory\(\)](#): frees different types of memory input : void pointer to an array, pointer to the size of the array and the type of the array. output : none.

4.9.3.5 void inc_refresh ()

function [inc_refresh\(\)](#): increments the time that we want the client to request data from server input : none. output : none.

4.9.3.6 gboolean init_timeout ()

function [init_timeout\(\)](#): sends a request to server and then waits for data,after it got all the data it inputs it in the right places and checks if the list_num_size is bigger then the LIST_SIZE if that is the case it removes the oldest element of the list and adds the newest to the begging.After the data has been properly handled it displays it in the lists and draws the new data on the graph.We check if the function is running in an infinite loop,if not we set it to run in regular intervals that we have set. input : none output : returns TRUE if we want to continue or FALSE if we want to stop;

4.9.3.7 void pause_app (GtkWidget * *button*)

4.9.3.8 void set_record (GtkWidget * *widget*)

function [set_record\(\)](#): sets the record flag to true or false depending on if the button is clicked or not input : widget. output : none.

4.9.3.9 int task_check (T_Collection * *tasks_new*, int *task_num*)

4.9.3.10 void test_strtol (long *val*)

4.9.3.11 void timeout_refresh ()

function [timeout_refresh\(\)](#): reruns the function init_timeout and tells the previous version to stop input : none. output : none.

4.9.4 Variable Documentation

4.9.4.1 Cpu_list* cpu_list

list to the interrupts

4.9.4.2 long cpu_num

4.9.4.3 bool* cpu_status

the size of the lists of cpu usage network usage and memory usage cant be bigger then LIST_SIZE

4.9.4.4 __int32_t dev_num_old

the space between the two data inputs number of devices

4.9.4.5 bool device_all

<bool used to check if the client wants all the devices shown

4.9.4.6 GtkWidget* device_swindow

>widget for creating the process window for editing the columns in the liststore for tasks

4.9.4.7 D_Collection* devices_old

4.9.4.8 GtkWidget* entry

4.9.4.9 long interrupt_num

4.9.4.10 I_Collection* interrupts

list to the tasks that we keep on client

4.9.4.11 I_Collection2* interrupts2

list to the interrupts

4.9.4.12 __int32_t list_num_size

number of tasks

4.9.4.13 Mega_Data* m_data

list to the memory usage

4.9.4.14 NetMem_list* mem_list

list to the network usage

4.9.4.15 NetMem_list* net_list

list to the cpu usage

4.9.4.16 int newsockfd**4.9.4.17 int newsockfd1****4.9.4.18 char p_dir[256]****4.9.4.19 GtkWidget* process_swindow**

>widget for creating the graph buttons window

4.9.4.20 bool record**4.9.4.21 guint refresh**

time interval for when the client requests data again

4.9.4.22 guint t**4.9.4.23 __int32_t task_num_old****4.9.4.24 T_Collection* tasks_old**

list to the devices that we keep on client

4.9.4.25 guint time_step

if the function init_timeout is in a loop this value is bigger then 0

4.9.4.26 GtkWidget* window_graphs**4.10 testing.c File Reference**

```
#include "testing.h"
#include "main_header.h"
#include <inttypes.h>
#include <memory.h>
#include <stdlib.h>
#include <errno.h>
```

Functions

- int [interrupts_write](#) ([I_Collection2](#) *array)
- int [memory_write](#) ([Memory_usage](#) *memory_usage)
- int [cpu_write](#) ([Cpu_usage](#) cpu_usage)
- int [cpu_read](#) ([Cpu_list](#) **array)
- int [task_write](#) ([T_Collection](#) *array)
- int [device_write](#) ([D_Collection](#) *array)
- int [netw_write](#) (uint64_t transmitted, uint64_t received)
- int [netw_calculate](#) (float *transmitted, float *received)
- int [ifstat_calculate](#) (float *received_kb, float *transmitted_kb)
- int [task_sort](#) ([T_Collection](#) **array, int number)

4.10.1 Function Documentation

4.10.1.1 int [cpu_read](#) ([Cpu_list](#) ** *array*)

4.10.1.2 int [cpu_write](#) ([Cpu_usage](#) *cpu_usage*)

4.10.1.3 int [device_write](#) ([D_Collection](#) * *array*)

4.10.1.4 int [ifstat_calculate](#) (float * *received_kb*, float * *transmitted_kb*)

4.10.1.5 int [interrupts_write](#) ([I_Collection2](#) * *array*)

4.10.1.6 int [memory_write](#) ([Memory_usage](#) * *memory_usage*)

4.10.1.7 int [netw_calculate](#) (float * *transmitted*, float * *received*)

4.10.1.8 int [netw_write](#) (uint64_t *transmitted*, uint64_t *received*)

4.10.1.9 int [task_sort](#) ([T_Collection](#) ** *array*, int *number*)

4.10.1.10 int [task_write](#) ([T_Collection](#) * *array*)

4.11 testing.h File Reference

```
#include "common.h"
#include "main_header.h"
```

Functions

- int [interrupts_write](#) ([I_Collection2](#) *array)
- int [cpu_write](#) ([Cpu_usage](#) cpu_usage)
- int [netw_write](#) (unsigned long transmitted, unsigned long received)
- int [memory_write](#) ([Memory_usage](#) *memory_usage)
- int [task_write](#) ([T_Collection](#) *array)
- int [device_write](#) ([D_Collection](#) *array)
- int [netw_calculate](#) (float *transmitted, float *received)
- int [ifstat_calculate](#) (float *received_kb, float *transmitted_kb)
- int [task_sort](#) ([T_Collection](#) **array, int number)
- int [cpu_read](#) ([Cpu_list](#) **array)

4.11.1 Function Documentation

4.11.1.1 int `cpu_read` (`Cpu_list` ** *array*)

4.11.1.2 int `cpu_write` (`Cpu_usage` *cpu_usage*)

4.11.1.3 int `device_write` (`D_Collection` * *array*)

4.11.1.4 int `ifstat_calculate` (float * *received_kb*, float * *transmitted_kb*)

4.11.1.5 int `interrupts_write` (`I_Collection2` * *array*)

4.11.1.6 int `memory_write` (`Memory_usage` * *memory_usage*)

4.11.1.7 int `netw_calculate` (float * *transmitted*, float * *received*)

4.11.1.8 int `netw_write` (unsigned long *transmitted*, unsigned long *received*)

4.11.1.9 int `task_sort` (`T_Collection` ** *array*, int *number*)

4.11.1.10 int `task_write` (`T_Collection` * *array*)

4.12 testing_tree.c File Reference

```
#include "testing_tree.h"
#include <errno.h>
#include "buttons.h"
#include "main_header.h"
```

Functions

- int `refresh_devices_data` (`D_Collection` **devices_new*, __int32_t *device_num*)
- void `delete_old_dev` (`D_Collection` ***array*, __int32_t **dev_num*)
- int `insert_new_devices` (`D_Collection` ***array*, `D_Collection` **devices_new*, __int32_t *dev_num*, __int32_t **old_number*)
- int `device_check` (`D_Collection` **devices_new*, int *dev_num*)
- int `insert_new_tasks` (`T_Collection` ***array*, `T_Collection` **tasks_new*, __int32_t *task_num*, __int32_t **old_number*)
- void `delete_old_tasks` (`T_Collection` ***array*, __int32_t **task_num*)
- int `refresh_task_data` (`T_Collection` **tasks_new*, int *task_num*)
- int `task_check` (`T_Collection` **tasks_new*, int *task_num*)
- void `create_list_store_task` (void)
- void `create_list_store_dev` (void)
- int `add_new_task` (`Task` **task_t*)
- int `add_new_dev` (`Devices` **devices*)
- void `change_list_store_view_devices` (`GtkWidget` **widget*)
- void `change_list_store_view_process` (`GtkWidget` **widget*)
- int `fill_task_item` (`Task` **task_item*, `GtkTreeIter` **iter*, int **array_i*)

- int [fill_device_item](#) ([Devices](#) *f_temp, GtkTreeIter *iter)
- void [refresh_list_item_device](#) ([Devices](#) *ref_temp)
- void [refresh_list_item](#) ([Task](#) *task_item, int *array_i)
- void [remove_task_item](#) (gint pid)
- void [remove_list_item_device](#) (gchar *directory, gchar *name)
- gint [compare_string_list_item](#) (GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)
- gint [compare_int_list_item](#) (GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)
- gint [compare_int_list_item_size](#) (GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)
- gint [compare_int_list_item_time](#) (GtkTreeModel *model, GtkTreeIter *iter1, GtkTreeIter *iter2, gpointer column)

4.12.1 Function Documentation

4.12.1.1 int [add_new_dev](#) ([Devices](#) * *devices*)

4.12.1.2 int [add_new_task](#) ([Task](#) * *task_t*)

4.12.1.3 void [change_list_store_view_devices](#) (GtkWidget * *widget*)

4.12.1.4 void [change_list_store_view_process](#) (GtkWidget * *widget*)

4.12.1.5 gint [compare_int_list_item](#) (GtkTreeModel * *model*, GtkTreeIter * *iter1*, GtkTreeIter * *iter2*, gpointer *column*)

4.12.1.6 gint [compare_int_list_item_size](#) (GtkTreeModel * *model*, GtkTreeIter * *iter1*, GtkTreeIter * *iter2*, gpointer *column*)

4.12.1.7 gint [compare_int_list_item_time](#) (GtkTreeModel * *model*, GtkTreeIter * *iter1*, GtkTreeIter * *iter2*, gpointer *column*)

4.12.1.8 gint [compare_string_list_item](#) (GtkTreeModel * *model*, GtkTreeIter * *iter1*, GtkTreeIter * *iter2*, gpointer *column*)

4.12.1.9 void [create_list_store_dev](#) (void)

4.12.1.10 void [create_list_store_task](#) (void)

4.12.1.11 void [delete_old_dev](#) ([D_Collection](#) ** *array*, __int32_t * *dev_num*)

4.12.1.12 void [delete_old_tasks](#) ([T_Collection](#) ** *array*, __int32_t * *task_num*)

4.12.1.13 int [device_check](#) ([D_Collection](#) * *devices_new*, int *dev_num*)

4.12.1.14 int [fill_device_item](#) ([Devices](#) * *f_temp*, GtkTreeIter * *iter*)

4.12.1.15 int [fill_task_item](#) ([Task](#) * *task_item*, GtkTreeIter * *iter*, int * *array_i*)

4.12.1.16 int [insert_new_devices](#) ([D_Collection](#) ** *array*, [D_Collection](#) * *devices_new*, __int32_t *dev_num*, __int32_t * *old_number*)

- 4.12.1.17 `int insert_new_tasks (T_Collection ** array, T_Collection * tasks_new, __int32_t task_num, __int32_t * old_number)`
- 4.12.1.18 `int refresh_devices_data (D_Collection * devices_new, __int32_t device_num)`
- 4.12.1.19 `void refresh_list_item (Task * task_item, int * array_i)`
- 4.12.1.20 `void refresh_list_item_device (Devices * ref_temp)`
- 4.12.1.21 `int refresh_task_data (T_Collection * tasks_new, int task_num)`
- 4.12.1.22 `void remove_list_item_device (gchar * directory, gchar * name)`
- 4.12.1.23 `void remove_task_item (gint pid)`
- 4.12.1.24 `int task_check (T_Collection * tasks_new, int task_num)`

4.13 testing_tree.h File Reference

```
#include "string.h"
#include <gtk/gtk.h>
#include <stdlib.h>
#include "common.h"
```

Enumerations

- enum {
COL_TASK = 0, COL_PID, COL_RSS, COL_CPU,
COL_PRIO, COL_VSZ, COL_PPID, COL_STATE,
COL_UNAME, COL_DUR, NUM_COLS }
- enum {
COL_DEV = 0, COL_DIR, COL_TYPE, COL_TOTAL,
COL_AVAILABLE, COL_USED, COL_FREE, NUM_COLS_DEV }

Functions

- `int task_check (T_Collection *tasks_new, int task_num)`
- `int device_check (D_Collection *devices_new, int dev_num)`
- `void delete_old_dev (D_Collection **array, __int32_t *dev_num)`
- `void delete_old_tasks (T_Collection **array, __int32_t *task_num)`
- `void remove_task_item (gint pid)`
- `void remove_list_item_device (gchar *directory, gchar *name)`
- `void refresh_list_item (Task *task_item, int *array_i)`
- `void refresh_list_item_device (Devices *ref_temp)`
- `int fill_task_item (Task *task_item, GtkTreeIter *iter, int *array_i)`
- `int fill_device_item (Devices *f_temp, GtkTreeIter *iter)`
- `void change_list_store_view_devices (GtkWidget *widget)`
- `void change_list_store_view_process (GtkWidget *widget)`

- int [add_new_task](#) (Task *task_t)
- int [add_new_dev](#) (Devices *devices)
- void [create_list_store_task](#) (void)
- gint [compare_string_list_item](#) (GtkTreeModel *model, GtkTreeliter *iter1, GtkTreeliter *iter2, gpointer column)
- gint [compare_int_list_item_size](#) (GtkTreeModel *model, GtkTreeliter *iter1, GtkTreeliter *iter2, gpointer column)
- gint [compare_int_list_item_time](#) (GtkTreeModel *model, GtkTreeliter *iter1, GtkTreeliter *iter2, gpointer column)
- gint [compare_int_list_item](#) (GtkTreeModel *model, GtkTreeliter *iter1, GtkTreeliter *iter2, gpointer column)
- void [create_list_store_dev](#) (void)

Variables

- GtkTreeSelection * [selection](#)
- GtkCellRenderer * [cell_renderer](#)
- GtkWidget * [treeview_tasks](#)
- GtkWidget * [treeview_devices](#)
- GtkTreeStore * [list_tasks](#)
- GtkTreeStore * [list_devices](#)

4.13.1 Enumeration Type Documentation

4.13.1.1 anonymous enum

Enumerator

COL_TASK
COL_PID
COL_RSS
COL_CPU
COL_PRIO
COL_VSZ
COL_PPID
COL_STATE
COL_UNAME
COL_DUR
NUM_COLS

4.13.1.2 anonymous enum

Enumerator

COL_DEV
COL_DIR
COL_TYPE
COL_TOTAL
COL_AVAILABLE
COL_USED
COL_FREE
NUM_COLS_DEV

4.13.2 Function Documentation

4.13.2.1 int add_new_dev (Devices * *devices*)

4.13.2.2 int add_new_task (Task * *task_t*)

4.13.2.3 void change_list_store_view_devices (GtkWidget * *widget*)

4.13.2.4 void change_list_store_view_process (GtkWidget * *widget*)

4.13.2.5 gint compare_int_list_item (GtkTreeModel * *model*, GtkTreeliter * *iter1*, GtkTreeliter * *iter2*, gpointer *column*)

4.13.2.6 gint compare_int_list_item_size (GtkTreeModel * *model*, GtkTreeliter * *iter1*, GtkTreeliter * *iter2*, gpointer *column*)

4.13.2.7 gint compare_int_list_item_time (GtkTreeModel * *model*, GtkTreeliter * *iter1*, GtkTreeliter * *iter2*, gpointer *column*)

4.13.2.8 gint compare_string_list_item (GtkTreeModel * *model*, GtkTreeliter * *iter1*, GtkTreeliter * *iter2*, gpointer *column*)

4.13.2.9 void create_list_store_dev (void)

4.13.2.10 void create_list_store_task (void)

4.13.2.11 void delete_old_dev (D_Collection ** *array*, __int32_t * *dev_num*)

4.13.2.12 void delete_old_tasks (T_Collection ** *array*, __int32_t * *task_num*)

4.13.2.13 int device_check (D_Collection * *devices_new*, int *dev_num*)

4.13.2.14 int fill_device_item (Devices * *f_temp*, GtkTreeliter * *iter*)

4.13.2.15 int fill_task_item (Task * *task_item*, GtkTreeliter * *iter*, int * *array_i*)

4.13.2.16 void refresh_list_item (Task * *task_item*, int * *array_i*)

4.13.2.17 void refresh_list_item_device (Devices * *ref_temp*)

4.13.2.18 void remove_list_item_device (gchar * *directory*, gchar * *name*)

4.13.2.19 void remove_task_item (gint *pid*)

4.13.2.20 int task_check (T_Collection * *tasks_new*, int *task_num*)

4.13.3 Variable Documentation

4.13.3.1 GtkWidget* *cell_renderer*

4.13.3.2 GtkWidget* *list_devices*

4.13.3.3 GtkWidget* *list_tasks*

treeview for devices

4.13.3.4 GtkWidget* selection

4.13.3.5 GtkWidget* treeview_devices

treeview for tasks

4.13.3.6 GtkWidget* treeview_tasks

4.14 window.c File Reference

```
#include "window.h"
#include "testing_tree.h"
#include "buttons.h"
#include "testing.h"
#include "drawing.h"
#include "functions.h"
```

Functions

- GtkWidget * [cpu_window](#) (int *cpu_number*, GtkWidget ***graph_list*)
- GtkWidget * [main_window](#) (GtkWidget **dev_swindow*, GtkWidget *[process_swindow](#))
- void [swap_change](#) ([Memory_usage](#) **memory_usage*)
- void [memory_change](#) ([Memory_usage](#) **memory_usage*)
- void [cpu_change](#) ([Cpu_usage](#) **cpu_usage*)
- void [network_change](#) ([Network](#) **network*)

4.14.1 Function Documentation

4.14.1.1 void [cpu_change](#) ([Cpu_usage](#) * *cpu_usage*)

4.14.1.2 GtkWidget* [cpu_window](#) (int *cpu_number*, GtkWidget ** *graph_list*)

4.14.1.3 GtkWidget* [main_window](#) (GtkWidget * *dev_swindow*, GtkWidget * *process_swindow*)

4.14.1.4 void [memory_change](#) ([Memory_usage](#) * *memory_usage*)

4.14.1.5 void [network_change](#) ([Network](#) * *network*)

function [network_change\(\)](#): inputs network usage into list and displays it textually in window input: pointer to Network usage output: none

4.14.1.6 void `swap_change` (`Memory_usage` * *memory_usage*)

4.15 window.h File Reference

```
#include "main_header.h"
#include "common.h"
```

Functions

- void `cpu_change` (`Cpu_usage` **cpu_usage*)
- void `memory_change` (`Memory_usage` **memory_usage*)
- void `swap_change` (`Memory_usage` **memory_usage*)
- void `network_change` (`Network` **network*)
- GtkWidget * `main_window` (GtkWidget **des_swindow*, GtkWidget **proc_swindow*)
- GtkWidget * `cpu_window` (int *cpu_number*, GtkWidget ***graph_list*)

Variables

- GtkWidget * `graph1`
- GtkWidget * `graph_net`
- GtkWidget * `graph_mem`
- GtkWidget * `graph_inttrp`
- GtkWidget * `graph_write`
- GtkWidget * `cpu_graphs`
- GtkWidget * `viewport`
- GtkAdjustment * `adj`
- GtkWidget * `interrupts_swindow`
- GtkWidget * `label_rec`
- GtkWidget * `label_trans`
- GtkWidget * `label_cpu0`
- GtkWidget * `label_mem`
- GtkWidget * `label_swap`
- GtkWidget * `CPU_WINDOW`

4.15.1 Function Documentation

4.15.1.1 void `cpu_change` (`Cpu_usage` * *cpu_usage*)

4.15.1.2 GtkWidget* `cpu_window` (int *cpu_number*, GtkWidget ** *graph_list*)

4.15.1.3 GtkWidget* `main_window` (GtkWidget * *des_swindow*, GtkWidget * *proc_swindow*)

4.15.1.4 void `memory_change` (`Memory_usage` * *memory_usage*)

4.15.1.5 void `network_change` (`Network` * *network*)

function `network_change()`: inputs network usage into list and displays it textually in window input: pointer to Network usage output: none

4.15.1.6 void swap_change (Memory_usage * *memory_usage*)

4.15.2 Variable Documentation

4.15.2.1 GtkAdjustment* adj

4.15.2.2 GtkWidget* cpu_graphs

4.15.2.3 GtkWidget* CPU_WINDOW

4.15.2.4 GtkWidget* graph1

4.15.2.5 GtkWidget* graph_inttrp

4.15.2.6 GtkWidget* graph_mem

4.15.2.7 GtkWidget* graph_net

4.15.2.8 GtkWidget* graph_write

4.15.2.9 GtkWidget* interrupts_swindow

4.15.2.10 GtkWidget* label_cpu0

4.15.2.11 GtkWidget* label_mem

4.15.2.12 GtkWidget* label_rec

4.15.2.13 GtkWidget* label_swap

4.15.2.14 GtkWidget* label_trans

4.15.2.15 GtkWidget* viewport

Index

- `__attribute__`
 - `common.h`, [19](#)
- `add_new_dev`
 - `testing_tree.c`, [36](#)
 - `testing_tree.h`, [39](#)
- `add_new_task`
 - `testing_tree.c`, [36](#)
 - `testing_tree.h`, [39](#)
- `adj`
 - `window.h`, [42](#)
- `app`
 - `main.c`, [28](#)
- `button_dec`
 - `buttons.h`, [15](#)
- `button_dev`
 - `buttons.h`, [15](#)
- `button_device_all`
 - `buttons.h`, [15](#)
- `button_device_avail`
 - `buttons.h`, [15](#)
- `button_device_devices`
 - `buttons.h`, [15](#)
- `button_device_directory`
 - `buttons.h`, [15](#)
- `button_device_free`
 - `buttons.h`, [15](#)
- `button_device_total`
 - `buttons.h`, [15](#)
- `button_device_type`
 - `buttons.h`, [15](#)
- `button_device_used`
 - `buttons.h`, [15](#)
- `button_graph`
 - `buttons.h`, [15](#)
- `button_inc`
 - `buttons.h`, [15](#)
- `button_pause`
 - `buttons.h`, [15](#)
- `button_proc`
 - `buttons.h`, [15](#)
- `button_process_cpu`
 - `buttons.h`, [15](#)
- `button_process_duration`
 - `buttons.h`, [15](#)
- `button_process_pid`
 - `buttons.h`, [15](#)
- `button_process_ppid`
 - `buttons.h`, [15](#)
- `button_process_prio`
 - `buttons.h`, [15](#)
- `button_process_rss`
 - `buttons.h`, [15](#)
- `button_process_state`
 - `buttons.h`, [16](#)
- `button_process_task`
 - `buttons.h`, [16](#)
- `button_process_user`
 - `buttons.h`, [16](#)
- `button_process_vm_size`
 - `buttons.h`, [16](#)
- `button_rec`
 - `buttons.h`, [16](#)
- `buttons.c`, [11](#)
 - `close_window`, [11](#)
 - `close_window_toggled`, [11](#)
 - `createTask_pop_up`, [11](#)
 - `device_window`, [12](#)
 - `graph_button_clicked`, [12](#)
 - `graph_clicked`, [12](#)
 - `handle_task_menu`, [12](#)
 - `handle_task_prio`, [12](#)
 - `on_treeview_tasks_button_press_event`, [12](#)
 - `process_window`, [12](#)
 - `record_window`, [12](#)
 - `show_all`, [12](#)
 - `show_hide`, [12](#)
- `buttons.h`, [12](#)
 - `button_dec`, [15](#)
 - `button_dev`, [15](#)
 - `button_device_all`, [15](#)
 - `button_device_avail`, [15](#)
 - `button_device_devices`, [15](#)
 - `button_device_directory`, [15](#)
 - `button_device_free`, [15](#)
 - `button_device_total`, [15](#)
 - `button_device_type`, [15](#)
 - `button_device_used`, [15](#)
 - `button_graph`, [15](#)
 - `button_inc`, [15](#)
 - `button_pause`, [15](#)
 - `button_proc`, [15](#)
 - `button_process_cpu`, [15](#)
 - `button_process_duration`, [15](#)
 - `button_process_pid`, [15](#)
 - `button_process_ppid`, [15](#)
 - `button_process_prio`, [15](#)
 - `button_process_rss`, [15](#)

- button_process_state, 16
- button_process_task, 16
- button_process_user, 16
- button_process_vm_size, 16
- button_rec, 16
- close_window, 14
- close_window_toggled, 14
- close_window_v1, 14
- closed_cpu_window, 16
- cpu_buttons, 16
- createTask_pop_up, 14
- dev_window, 16
- device_window, 14
- graph_button_clicked, 14
- graph_clicked, 14
- handle_task_menu, 14
- handle_task_prio, 14
- on_treeview_tasks_button_press_event, 14
- proc_window, 16
- process_window, 14
- rec_window, 16
- record_window, 14
- show_all, 14
- show_hide, 15
- task_popup, 16
- wr_window, 16
- write_window, 15
- COL_AVAILABLE
 - testing_tree.h, 38
- COL_CPU
 - testing_tree.h, 38
- COL_DEV
 - testing_tree.h, 38
- COL_DIR
 - testing_tree.h, 38
- COL_DUR
 - testing_tree.h, 38
- COL_FREE
 - testing_tree.h, 38
- COL_PID
 - testing_tree.h, 38
- COL_PPID
 - testing_tree.h, 38
- COL_PRIO
 - testing_tree.h, 38
- COL_RSS
 - testing_tree.h, 38
- COL_STATE
 - testing_tree.h, 38
- COL_TASK
 - testing_tree.h, 38
- COL_TOTAL
 - testing_tree.h, 38
- COL_TYPE
 - testing_tree.h, 38
- COL_UNAME
 - testing_tree.h, 38
- COL_USED
 - testing_tree.h, 38
- COL_VSZ
 - testing_tree.h, 38
- CPU_PACK
 - common.h, 17
- CPU_USAGE
 - common.h, 17
- CPU_WINDOW
 - window.h, 42
- cell_renderer
 - testing_tree.h, 39
- change_list_store_view_devices
 - testing_tree.c, 36
 - testing_tree.h, 39
- change_list_store_view_process
 - testing_tree.c, 36
 - testing_tree.h, 39
- checking_interrupt_names
 - drawing.h, 22
- close_window
 - buttons.c, 11
 - buttons.h, 14
- close_window_toggled
 - buttons.c, 11
 - buttons.h, 14
- close_window_v1
 - buttons.h, 14
- closed_cpu_window
 - buttons.h, 16
- command_sender
 - functions.c, 24
 - functions.h, 25
- common.h, 16
 - __attribute__, 19
 - CPU_PACK, 17
 - CPU_USAGE, 17
 - Cpu_usage, 18
 - D_Collection, 18
 - DEVICES, 17
 - Data, 18
 - Devices, 18
 - fontdesc, 19
 - I_Collection, 18
 - I_Collection2, 18
 - INT_PACK, 18
 - INTERRUPTS, 18
 - Interrupts, 18
 - Interrupts2, 18
 - Interrupts_send, 18
 - MEMORY, 18
 - Memory_usage, 18
 - NETWORK, 18
 - Network, 18
 - T_Collection, 18
 - TASK, 18
 - TEXT, 18
 - Task, 18
 - Unification, 19

- compare_int_list_item
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- compare_int_list_item_size
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- compare_int_list_item_time
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- compare_string_list_item
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- conformation
 - Unification, [10](#)
- connection
 - functions.c, [24](#)
 - functions.h, [25](#)
- Cpu_List, [5](#)
 - data, [5](#)
 - next, [5](#)
- cpu_buttons
 - buttons.h, [16](#)
- cpu_change
 - window.c, [40](#)
 - window.h, [41](#)
- cpu_graphs
 - window.h, [42](#)
- Cpu_list
 - main_header.h, [30](#)
- cpu_list
 - main_header.h, [31](#)
- cpu_num
 - main_header.h, [31](#)
- cpu_read
 - testing.c, [34](#)
 - testing.h, [35](#)
- cpu_stats
 - Mega_Data, [8](#)
- cpu_status
 - main_header.h, [32](#)
- Cpu_usage
 - common.h, [18](#)
- cpu_window
 - window.c, [40](#)
 - window.h, [41](#)
- cpu_write
 - testing.c, [34](#)
 - testing.h, [35](#)
- create_list_store_dev
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- create_list_store_task
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- createTask_pop_up
 - buttons.c, [11](#)
 - buttons.h, [14](#)
- D_Collection
 - common.h, [18](#)
- DEVICES
 - common.h, [17](#)
- Data
 - common.h, [18](#)
- data
 - Cpu_List, [5](#)
 - NetMem_list, [8](#)
- data_pack
 - Unification, [10](#)
- data_transfer
 - functions.c, [24](#)
 - functions.h, [25](#)
- dec_refresh
 - main.c, [27](#)
 - main_header.h, [30](#)
- delete_old_dev
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- delete_old_tasks
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- destroy_window
 - main.c, [27](#)
 - main_header.h, [30](#)
- dev_num_old
 - main_header.h, [32](#)
- dev_window
 - buttons.h, [16](#)
- Device_Collection, [5](#)
 - devices, [6](#)
 - next, [6](#)
 - prev, [6](#)
- device_all
 - main_header.h, [32](#)
- device_check
 - main_header.h, [31](#)
 - testing_tree.c, [36](#)
 - testing_tree.h, [39](#)
- device_list
 - Mega_Data, [8](#)
- device_swindow
 - main_header.h, [32](#)
- device_task_commands
 - functions.c, [24](#)
 - functions.h, [25](#)
- device_window
 - buttons.c, [12](#)
 - buttons.h, [14](#)
- device_write
 - testing.c, [34](#)
 - testing.h, [35](#)
- Devices
 - common.h, [18](#)
- devices
 - Device_Collection, [6](#)
 - Unification, [10](#)
- devices_old

- main_header.h, 32
- do_drawing_cpu
 - drawing.c, 20
 - drawing.h, 22
- do_drawing_cpu2
 - drawing.h, 22
- do_drawing_int
 - drawing.h, 22
- do_drawing_int2
 - drawing.c, 20
 - drawing.h, 22
- do_drawing_mem
 - drawing.c, 20
 - drawing.h, 22
- do_drawing_net
 - drawing.c, 20
 - drawing.h, 22
- do_drawing_one_cpu
 - drawing.c, 20
 - drawing.h, 22
- draw_frame
 - drawing.c, 20
 - drawing.h, 22
- draw_graph
 - drawing.c, 20
 - drawing.h, 22
- draw_graph_mem
 - drawing.c, 20
 - drawing.h, 22
- draw_graph_net
 - drawing.c, 20
 - drawing.h, 22
- draw_interrupts
 - drawing.h, 23
- draw_interrupts2
 - drawing.c, 20
- draw_percentages
 - drawing.c, 21
 - drawing.h, 23
- drawing.c, 19
 - do_drawing_cpu, 20
 - do_drawing_int2, 20
 - do_drawing_mem, 20
 - do_drawing_net, 20
 - do_drawing_one_cpu, 20
 - draw_frame, 20
 - draw_graph, 20
 - draw_graph_mem, 20
 - draw_graph_net, 20
 - draw_interrupts2, 20
 - draw_percentages, 21
 - on_draw_event, 21
 - writing_interrupt_names2, 21
 - writing_seconds, 21
- drawing.h, 21
 - checking_interrupt_names, 22
 - do_drawing_cpu, 22
 - do_drawing_cpu2, 22
 - do_drawing_int, 22
 - do_drawing_int2, 22
 - do_drawing_mem, 22
 - do_drawing_net, 22
 - do_drawing_one_cpu, 22
 - draw_frame, 22
 - draw_graph, 22
 - draw_graph_mem, 22
 - draw_graph_net, 22
 - draw_interrupts, 23
 - draw_percentages, 23
 - on_draw_event, 23
 - on_draw_event2, 23
 - writing_interrupt_names, 23
 - writing_interrupt_names2, 23
 - writing_seconds, 23
- entry
 - main_header.h, 32
- fill_device_item
 - testing_tree.c, 36
 - testing_tree.h, 39
- fill_task_item
 - testing_tree.c, 36
 - testing_tree.h, 39
- flag_timeout
 - main.c, 28
- fontdesc
 - common.h, 19
- free_mega_data
 - main.c, 27
- free_one_mega_data
 - main.c, 27
- freeing_memory
 - main.c, 27
 - main_header.h, 31
- functions.c, 23
 - command_sender, 24
 - connection, 24
 - data_transfer, 24
 - device_task_commands, 24
 - input_command, 24
 - receive_number_cpu, 24
 - scan_numbers, 24
 - test_recv, 24
 - test_send, 24
- functions.h, 25
 - command_sender, 25
 - connection, 25
 - data_transfer, 25
 - device_task_commands, 25
 - input_command, 25
 - receive_number_cpu, 26
 - test_recv, 26
 - test_send, 26
- graph1
 - window.h, 42

- graph_button_clicked
 - buttons.c, [12](#)
 - buttons.h, [14](#)
- graph_clicked
 - buttons.c, [12](#)
 - buttons.h, [14](#)
- graph_inttrp
 - window.h, [42](#)
- graph_mem
 - window.h, [42](#)
- graph_net
 - window.h, [42](#)
- graph_write
 - window.h, [42](#)
- handle_task_menu
 - buttons.c, [12](#)
 - buttons.h, [14](#)
- handle_task_prio
 - buttons.c, [12](#)
 - buttons.h, [14](#)
- I_Collection
 - common.h, [18](#)
- I_Collection2
 - common.h, [18](#)
- INT_PACK
 - common.h, [18](#)
- INTERRUPTS
 - common.h, [18](#)
- ifstat_calculate
 - testing.c, [34](#)
 - testing.h, [35](#)
- inc_refresh
 - main.c, [27](#)
 - main_header.h, [31](#)
- init_timeout
 - main.c, [27](#)
 - main_header.h, [31](#)
- input_command
 - functions.c, [24](#)
 - functions.h, [25](#)
- insert_new_devices
 - testing_tree.c, [36](#)
- insert_new_tasks
 - testing_tree.c, [36](#)
- Interrupt_Collection, [6](#)
 - interrupts, [6](#)
 - next, [6](#)
 - prev, [6](#)
- Interrupt_Collection2, [7](#)
 - interrupts, [7](#)
 - next, [7](#)
 - prev, [7](#)
- interrupt_num
 - main_header.h, [32](#)
- Interrupts
 - common.h, [18](#)
- interrupts
 - Interrupt_Collection, [6](#)
 - Interrupt_Collection2, [7](#)
 - main_header.h, [32](#)
 - Unification, [10](#)
- Interrupts2
 - common.h, [18](#)
- interrupts2
 - main_header.h, [32](#)
- interrupts_list
 - Mega_Data, [8](#)
- Interrupts_send
 - common.h, [18](#)
- interrupts_send
 - Unification, [10](#)
- interrupts_swindow
 - window.h, [42](#)
- interrupts_write
 - testing.c, [34](#)
 - testing.h, [35](#)
- LIST_SIZE
 - main_header.h, [30](#)
- label_cpu0
 - window.h, [42](#)
- label_mem
 - window.h, [42](#)
- label_rec
 - window.h, [42](#)
- label_swap
 - window.h, [42](#)
- label_trans
 - window.h, [42](#)
- list_devices
 - testing_tree.h, [39](#)
- list_num_size
 - main_header.h, [32](#)
- list_tasks
 - testing_tree.h, [39](#)
- m_data
 - main_header.h, [32](#)
- MEMORY
 - common.h, [18](#)
- main
 - main.c, [27](#)
- main.c, [26](#)
 - app, [28](#)
 - dec_refresh, [27](#)
 - destroy_window, [27](#)
 - flag_timeout, [28](#)
 - free_mega_data, [27](#)
 - free_one_mega_data, [27](#)
 - freeing_memory, [27](#)
 - inc_refresh, [27](#)
 - init_timeout, [27](#)
 - main, [27](#)
 - pause_app, [28](#)
 - semt, [28](#)
 - set_record, [28](#)

- test_strtol, 28
- timeout_refresh, 28
- window, 28
- writing, 28
- main_header.h, 29
 - Cpu_list, 30
 - cpu_list, 31
 - cpu_num, 31
 - cpu_status, 32
 - dec_refresh, 30
 - destroy_window, 30
 - dev_num_old, 32
 - device_all, 32
 - device_check, 31
 - device_swindow, 32
 - devices_old, 32
 - entry, 32
 - freeing_memory, 31
 - inc_refresh, 31
 - init_timeout, 31
 - interrupt_num, 32
 - interrupts, 32
 - interrupts2, 32
 - LIST_SIZE, 30
 - list_num_size, 32
 - m_data, 32
 - Mega_Data, 30
 - mem_list, 32
 - net_list, 33
 - NetMem_list, 30
 - newsockfd, 33
 - newsockfd1, 33
 - p_dir, 33
 - pause_app, 31
 - process_swindow, 33
 - record, 33
 - refresh, 33
 - set_record, 31
 - t, 33
 - task_check, 31
 - task_num_old, 33
 - tasks_old, 33
 - test_strtol, 31
 - time_step, 33
 - timeout_refresh, 31
 - window_graphs, 33
- main_window
 - window.c, 40
 - window.h, 41
- Mega_Data, 7
 - cpu_stats, 8
 - device_list, 8
 - interrupts_list, 8
 - main_header.h, 30
 - mem_stats, 8
 - net_stats, 8
 - next, 8
 - task_list, 8
- mem_list
 - main_header.h, 32
- mem_stats
 - Mega_Data, 8
- memory_change
 - window.c, 40
 - window.h, 41
- Memory_usage
 - common.h, 18
- memory_usage
 - Unification, 10
- memory_write
 - testing.c, 34
 - testing.h, 35
- NETWORK
 - common.h, 18
- NUM_COLS_DEV
 - testing_tree.h, 38
- NUM_COLS
 - testing_tree.h, 38
- net_list
 - main_header.h, 33
- net_stats
 - Mega_Data, 8
- NetMem_list, 8
 - data, 8
 - main_header.h, 30
 - next, 8
- netw_calculate
 - testing.c, 34
 - testing.h, 35
- netw_write
 - testing.c, 34
 - testing.h, 35
- Network
 - common.h, 18
- network
 - Unification, 10
- network_change
 - window.c, 40
 - window.h, 41
- newsockfd
 - main_header.h, 33
- newsockfd1
 - main_header.h, 33
- next
 - Cpu_List, 5
 - Device_Collection, 6
 - Interrupt_Collection, 6
 - Interrupt_Collection2, 7
 - Mega_Data, 8
 - NetMem_list, 8
 - Task_Collection, 9
- on_draw_event
 - drawing.c, 21
 - drawing.h, 23
- on_draw_event2

- drawing.h, 23
- on_treeview_tasks_button_press_event
 - buttons.c, 12
 - buttons.h, 14
- p_dir
 - main_header.h, 33
- pause_app
 - main.c, 28
 - main_header.h, 31
- prev
 - Device_Collection, 6
 - Interrupt_Collection, 6
 - Interrupt_Collection2, 7
 - Task_Collection, 9
- proc_window
 - buttons.h, 16
- process_swindow
 - main_header.h, 33
- process_window
 - buttons.c, 12
 - buttons.h, 14
- rec_window
 - buttons.h, 16
- receive_number_cpu
 - functions.c, 24
 - functions.h, 26
- record
 - main_header.h, 33
- record_window
 - buttons.c, 12
 - buttons.h, 14
- refresh
 - main_header.h, 33
- refresh_devices_data
 - testing_tree.c, 37
- refresh_list_item
 - testing_tree.c, 37
 - testing_tree.h, 39
- refresh_list_item_device
 - testing_tree.c, 37
 - testing_tree.h, 39
- refresh_task_data
 - testing_tree.c, 37
- remove_list_item_device
 - testing_tree.c, 37
 - testing_tree.h, 39
- remove_task_item
 - testing_tree.c, 37
 - testing_tree.h, 39
- scan_numbers
 - functions.c, 24
- selection
 - testing_tree.h, 39
- sem_t
 - main.c, 28
- set_record
 - main.c, 28
 - main_header.h, 31
- show_all
 - buttons.c, 12
 - buttons.h, 14
- show_hide
 - buttons.c, 12
 - buttons.h, 15
- swap_change
 - window.c, 40
 - window.h, 41
- t
 - main_header.h, 33
- T_Collection
 - common.h, 18
- TASK
 - common.h, 18
- TEXT
 - common.h, 18
- Task
 - common.h, 18
- task
 - Task_Collection, 9
 - Unification, 10
- Task_Collection, 9
 - next, 9
 - prev, 9
 - task, 9
- task_check
 - main_header.h, 31
 - testing_tree.c, 37
 - testing_tree.h, 39
- task_list
 - Mega_Data, 8
- task_num_old
 - main_header.h, 33
- task_popup
 - buttons.h, 16
- task_sort
 - testing.c, 34
 - testing.h, 35
- task_write
 - testing.c, 34
 - testing.h, 35
- tasks_old
 - main_header.h, 33
- test_recv
 - functions.c, 24
 - functions.h, 26
- test_send
 - functions.c, 24
 - functions.h, 26
- test_strtol
 - main.c, 28
 - main_header.h, 31
- testing.c, 33
 - cpu_read, 34
 - cpu_write, 34

- device_write, 34
- ifstat_calculate, 34
- interrupts_write, 34
- memory_write, 34
- netw_calculate, 34
- netw_write, 34
- task_sort, 34
- task_write, 34
- testing.h, 34
 - cpu_read, 35
 - cpu_write, 35
 - device_write, 35
 - ifstat_calculate, 35
 - interrupts_write, 35
 - memory_write, 35
 - netw_calculate, 35
 - netw_write, 35
 - task_sort, 35
 - task_write, 35
- testing_tree.c, 35
 - add_new_dev, 36
 - add_new_task, 36
 - change_list_store_view_devices, 36
 - change_list_store_view_process, 36
 - compare_int_list_item, 36
 - compare_int_list_item_size, 36
 - compare_int_list_item_time, 36
 - compare_string_list_item, 36
 - create_list_store_dev, 36
 - create_list_store_task, 36
 - delete_old_dev, 36
 - delete_old_tasks, 36
 - device_check, 36
 - fill_device_item, 36
 - fill_task_item, 36
 - insert_new_devices, 36
 - insert_new_tasks, 36
 - refresh_devices_data, 37
 - refresh_list_item, 37
 - refresh_list_item_device, 37
 - refresh_task_data, 37
 - remove_list_item_device, 37
 - remove_task_item, 37
 - task_check, 37
- testing_tree.h, 37
 - add_new_dev, 39
 - add_new_task, 39
 - COL_AVAILABLE, 38
 - COL_CPU, 38
 - COL_DEV, 38
 - COL_DIR, 38
 - COL_DUR, 38
 - COL_FREE, 38
 - COL_PID, 38
 - COL_PPID, 38
 - COL_PRIO, 38
 - COL_RSS, 38
 - COL_STATE, 38
 - COL_TASK, 38
 - COL_TOTAL, 38
 - COL_TYPE, 38
 - COL_UNAME, 38
 - COL_USED, 38
 - COL_VSZ, 38
 - cell_renderer, 39
 - change_list_store_view_devices, 39
 - change_list_store_view_process, 39
 - compare_int_list_item, 39
 - compare_int_list_item_size, 39
 - compare_int_list_item_time, 39
 - compare_string_list_item, 39
 - create_list_store_dev, 39
 - create_list_store_task, 39
 - delete_old_dev, 39
 - delete_old_tasks, 39
 - device_check, 39
 - fill_device_item, 39
 - fill_task_item, 39
 - list_devices, 39
 - list_tasks, 39
 - NUM_COLS_DEV, 38
 - NUM_COLS, 38
 - refresh_list_item, 39
 - refresh_list_item_device, 39
 - remove_list_item_device, 39
 - remove_task_item, 39
 - selection, 39
 - task_check, 39
 - treeview_devices, 40
 - treeview_tasks, 40
- time_step
 - main_header.h, 33
- timeout_refresh
 - main.c, 28
 - main_header.h, 31
- treeview_devices
 - testing_tree.h, 40
- treeview_tasks
 - testing_tree.h, 40
- Unification, 9
 - common.h, 19
 - conformation, 10
 - data_pack, 10
 - devices, 10
 - interrupts, 10
 - interrupts_send, 10
 - memory_usage, 10
 - network, 10
 - task, 10
- viewport
 - window.h, 42
- window
 - main.c, 28
- window.c, 40

- cpu_change, [40](#)
- cpu_window, [40](#)
- main_window, [40](#)
- memory_change, [40](#)
- network_change, [40](#)
- swap_change, [40](#)
- window.h, [41](#)
 - adj, [42](#)
 - CPU_WINDOW, [42](#)
 - cpu_change, [41](#)
 - cpu_graphs, [42](#)
 - cpu_window, [41](#)
 - graph1, [42](#)
 - graph_inttrp, [42](#)
 - graph_mem, [42](#)
 - graph_net, [42](#)
 - graph_write, [42](#)
 - interrupts_swindow, [42](#)
 - label_cpu0, [42](#)
 - label_mem, [42](#)
 - label_rec, [42](#)
 - label_swap, [42](#)
 - label_trans, [42](#)
 - main_window, [41](#)
 - memory_change, [41](#)
 - network_change, [41](#)
 - swap_change, [41](#)
 - viewport, [42](#)
- window_graphs
 - main_header.h, [33](#)
- wr_window
 - buttons.h, [16](#)
- write_window
 - buttons.h, [15](#)
- writing
 - main.c, [28](#)
- writing_interrupt_names
 - drawing.h, [23](#)
- writing_interrupt_names2
 - drawing.c, [21](#)
 - drawing.h, [23](#)
- writing_seconds
 - drawing.c, [21](#)
 - drawing.h, [23](#)