

AOSV PROJECT

Generated by Doxygen 1.8.17

1 AOSV Final Project Sources	1
1.1 <tt>Headers/	1
1.2 <tt>Module/	1
1.3 <tt>./	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 args Struct Reference	7
4.1.1 Detailed Description	7
4.2 completion_list Struct Reference	7
4.2.1 Detailed Description	8
4.2.2 Field Documentation	8
4.2.2.1 id	8
4.2.2.2 list_lock	8
4.2.2.3 sem_counter	8
4.2.2.4 to_destroy	8
4.2.2.5 workers	9
4.3 completion_list_data Struct Reference	9
4.3.1 Detailed Description	9
4.3.2 Field Documentation	10
4.3.2.1 head	10
4.3.2.2 id	10
4.3.2.3 requested_size	10
4.4 dequeue_fast_list Struct Reference	11
4.4.1 Detailed Description	11
4.4.2 Field Documentation	11
4.4.2.1 cur_size	11
4.4.2.2 list_data	12
4.4.2.3 list_pointer	12
4.4.2.4 origin_size	12
4.5 sched_thread_data Struct Reference	12
4.5.1 Detailed Description	13
4.5.2 Field Documentation	13
4.5.2.1 args	13
4.5.2.2 completion_list	13
4.5.2.3 entry_point	13
4.6 scheduler_thread Struct Reference	14
4.6.1 Detailed Description	14

4.6.2 Field Documentation	15
4.6.2.1 args	15
4.6.2.2 currently_running	15
4.6.2.3 destroy	15
4.6.2.4 ended	15
4.6.2.5 ent	15
4.6.2.6 ent_write	16
4.6.2.7 entry_point	16
4.6.2.8 father	16
4.6.2.9 last_switch_time	16
4.6.2.10 mean_switch_time	16
4.6.2.11 nesting	17
4.6.2.12 numb_switch	17
4.6.2.13 pid	17
4.6.2.14 started	17
4.6.2.15 yield_priority	17
4.7 scheduler_thread_worker_lifo Struct Reference	18
4.7.1 Detailed Description	18
4.8 ums_device_data_s Struct Reference	18
4.8.1 Detailed Description	19
4.9 UMS_sched_data Struct Reference	19
4.9.1 Detailed Description	19
4.9.2 Field Documentation	19
4.9.2.1 ent	19
4.9.2.2 ent_write	20
4.9.2.3 father	20
4.9.2.4 id_counter	20
4.9.2.5 owner	20
4.9.2.6 scheduler_thread	20
4.10 ums_worker_thread_data Struct Reference	21
4.10.1 Detailed Description	21
4.10.2 Field Documentation	21
4.10.2.1 args	21
4.10.2.2 completion_list_id	22
4.10.2.3 next	22
4.10.2.4 pid	22
4.10.2.5 work	22
4.11 worker_thread Struct Reference	22
4.11.1 Detailed Description	23
4.11.2 Field Documentation	23
4.11.2.1 args	23
4.11.2.2 completion_list_id	23

4.11.2.3 execute_lock	23
4.11.2.4 numb_switch	24
4.11.2.5 pid	24
4.11.2.6 state	24
4.11.2.7 task_struct	24
4.11.2.8 work	24
5 File Documentation	25
5.1 Headers/Common.h File Reference	25
5.1.1 Detailed Description	26
5.1.2 Macro Definition Documentation	26
5.1.2.1 ADD_WORKER	26
5.1.2.2 CREATE_COMP_LIST	27
5.1.2.3 DEBUG_DO_PPRINTK	28
5.1.2.4 DEBUG_DO_PRINTK	28
5.1.2.5 DEQUEUE_SIZE_REQUEST	28
5.1.2.6 DEQUEUE_UMS_COMPLETION_LIST_ITEMS	29
5.1.2.7 DESTROY_COMP_LIST	29
5.1.2.8 ENTER_UMS_SCHEDULING_MODE	30
5.1.2.9 ERR_PPRINTK	30
5.1.2.10 ERR_PRINTK	30
5.1.2.11 EXECUTE_UMS_THREAD	30
5.1.2.12 EXIT_FROM_YIELD	31
5.1.2.13 EXIT_WORKER_THREAD	31
5.1.2.14 NOTIFY_EXEC_ERR_PPRINTK	32
5.1.2.15 RELEASE_UMS	32
5.1.2.16 UMS_THREAD_YIELD	32
5.2 Headers/def_struct.h File Reference	33
5.2.1 Detailed Description	35
5.2.2 Typedef Documentation	35
5.2.2.1 completion_list	35
5.2.2.2 scheduler_thread	35
5.2.2.3 scheduler_thread_worker_lifo	35
5.2.2.4 UMS_sched_data	36
5.2.2.5 worker_thread	36
5.3 Headers/export_proc_info.h File Reference	36
5.3.1 Detailed Description	37
5.3.2 Function Documentation	37
5.3.2.1 create_proc_ums()	37
5.3.2.2 create_proc_ums_scheduler()	38
5.3.2.3 create_proc_ums_scheduler_thread()	38
5.3.2.4 create_proc_worker()	39

5.3.2.5 rm_proc_ums()	39
5.3.2.6 rm_proc_ums_scheduler()	39
5.3.2.7 rm_proc_ums_scheduler_thread()	40
5.3.2.8 rm_proc_worker()	40
5.4 Headers/ums_user_library.h File Reference	41
5.4.1 Detailed Description	42
5.4.2 Macro Definition Documentation	42
5.4.2.1 DEBUG_USER_DO_PPRINTF	43
5.4.2.2 DEBUG_USER_DO_PRINTF	43
5.4.2.3 ERR_USER_EXEC_PPRINTF	43
5.4.2.4 ERR_USER_EXEC_PRINTF	44
5.4.2.5 ERR_USER_PPRINTF	44
5.4.2.6 ERR_USER_PRINTF	44
5.4.3 Function Documentation	44
5.4.3.1 add_worker_thread()	44
5.4.3.2 create_completion_list()	45
5.4.3.3 dequeue_ums_completion_list_items()	45
5.4.3.4 destroy_comp_list()	46
5.4.3.5 enter_ums_scheduling_mode()	46
5.4.3.6 execute_ums_thread()	47
5.4.3.7 exit_worker_thread()	47
5.4.3.8 release_ums()	48
5.4.3.9 ums_close()	48
5.4.3.10 ums_init()	49
5.4.3.11 ums_thread_yield()	49
5.5 Headers/user_struct.h File Reference	49
5.5.1 Detailed Description	50
5.5.2 Typedef Documentation	50
5.5.2.1 completion_list_data	51
5.5.2.2 sched_thread_data	51
5.5.2.3 ums_worker_thread_data	51
5.6 Module/module_library.h File Reference	51
5.6.1 Detailed Description	53
5.6.2 Function Documentation	53
5.6.2.1 _execute_ums_thread()	53
5.6.2.2 _retrive_scheduler_thread_proc()	54
5.6.2.3 _retrive_worker_thread_proc()	54
5.6.2.4 add_worker()	55
5.6.2.5 create_ums_thread()	56
5.6.2.6 del_worker_thread()	56
5.6.2.7 dequeue_size_request()	57
5.6.2.8 destroy_comp_list_set_destroy()	57

5.6.2.9 enter_ums_scheduling_mode()	58
5.6.2.10 execute_ums_thread()	58
5.6.2.11 exit_from_yield()	59
5.6.2.12 exit_worker_thread()	59
5.6.2.13 list_copy_to_user()	60
5.6.2.14 release_scheduler()	60
5.6.2.15 retrieve_current_list()	61
5.6.2.16 retrieve_current_scheduler_thread()	61
5.6.2.17 retrieve_list()	62
5.6.2.18 retrieve_worker_thread()	63
5.6.2.19 ums_thread_yield()	63
5.7 Module/ums.h File Reference	63
5.7.1 Detailed Description	65
5.7.2 Function Documentation	65
5.7.2.1 retrieve_scheduler_thread_proc()	65
5.7.2.2 retrieve_worker_thread_proc()	65
5.7.2.3 ums_dev_ioctl()	66
5.7.2.4 ums_dev_open()	67
5.7.2.5 ums_dev_release()	67
Index	69

Chapter 1

AOSV Final Project Sources

A.Y. 2020/2021

Author(s): Daniele De Turris (1919828), Francesco Douglas Scotti di Vigoleno (1743635)

The sources are structured in this way:

1.1 **<tt>Headers/</tt>** cointains both header files and libraries for both user and kernel space

- **Commonn.h**: ioctl's definition
- **def_struct.h** (p. ??): struct used by the LKM
- **export_porc_info.h/export_porc_info.c**: code used by the LKM to manage /proc/ums
- **ums_user_library.h** (p. ??)/**ums_user_library.c**: library to interct with LKM from user space
- **user_struct.h** (p. ??): structs definiton used by **ums_user_library.h** (p. ??) to exchnage infos with the LKM

1.2 **<tt>Module/</tt>** cointains the code of the LKM and its library

- **ums.c** (p. ??)/**ums.h**: LKM code
- **module_library.c** (p. ??)/**module_library.h**: support library for LKM

1.3 **<tt>./</tt>** cointains the test file

- **test.c** (p. ??): test file

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

args	7
completion_list Structure used by an UMS Scheduler to maintain Completion list infos, used in CREATE_CO↔ MP_LIST	7
completion_list_data Struct used to share completion_list_data (p. ??) info with the kernel	9
dequeue_fast_list Struct used by a UMS Scheduler thread to maintain a pre-allocated completion_list_data* used in the dequeue function	11
sched_thread_data Struct used to share sched_thread_data (p. ??) info with the kernel	12
scheduler_thread Structure used by an UMS Scheduler to maintain scheduler thread infos, used in ENTER_SC↔ HEDULING_MODE	14
scheduler_thread_worker_lifo Strucute used by a UMS Scheduler thread to save the order of the worker threads called . . .	18
ums_device_data_s Structure containing the data needed by module to initialize the UMS device driver	18
UMS_sched_data Structure used by the kernel module to maintain differrent UMS Scheduler	19
ums_worker_thread_data Struct used to share ums_worker_thread_data (p. ??) infos with the kernel	21
worker_thread Strucute used by a UMS Scheduler to maintain Worker Thread info, used in ADD_WORKER . .	22

Chapter 3

File Index

3.1 File List

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

test.c	??
Headers/ Common.h	
Contains constants and macros used both by the module and the user library	25
Headers/ def_struct.h	
Contains the definitions of every custom type used in the Kernel module code	33
Headers/ export_proc_info.c	??
Headers/ export_proc_info.h	
This file contains, macros and functions used to manage the /proc files	36
Headers/ ums_user_library.c	??
Headers/ ums_user_library.h	
This library contains the definitions of every function that can be used from user space to interact with the UMS Scheduler device	41
Headers/ user_struct.h	
Contains the definitions of every custom type used in user space	49
Module/ module_library.c	??
Module/ module_library.h	
Contains the helper functions called from the UMS device	51
Module/ ums.c	??
Module/ ums.h	
Contains the functions to interact from user space with the kernel module	63

Chapter 4

Data Structure Documentation

4.1 args Struct Reference

Data Fields

- int **fd**
- int **i**
- ums_pid_t **comp_list_id**
- int **padre**

4.1.1 Detailed Description

Definition at line 10 of file test.c.

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

- test.c

4.2 completion_list Struct Reference

Structure used by an UMS Scheduler to mantain Completion list infos, used in CREATE_COMP_LIST.

```
#include <def_struct.h>
```

Data Fields

- ums_pid_t **id**
- unsigned char **to_destroy**
- spinlock_t **list_lock**
- struct semaphore **sem_counter**
- struct list_head **next**
- struct list_head **workers**

4.2.1 Detailed Description

Structure used by an UMS Scheduler to maintain Completion list infos, used in CREATE_COMP_LIST.

Definition at line 67 of file def_struct.h.

4.2.2 Field Documentation

4.2.2.1 id

```
ums_pid_t id
```

id of current completion list

Definition at line 69 of file def_struct.h.

4.2.2.2 list_lock

```
spinlock_t list_lock
```

used for concurrency while add new worker

Definition at line 71 of file def_struct.h.

4.2.2.3 sem_counter

```
struct semaphore sem_counter
```

used by a scheduler thread to be sure that some worker can be executed

Definition at line 72 of file def_struct.h.

4.2.2.4 to_destroy

```
unsigned char to_destroy
```

bool used by DESTROY_COMP_LIST to denie new worker insertion

Definition at line 70 of file def_struct.h.

4.2.2.5 workers

```
struct list_head workers
```

reference to the worker threads list

Definition at line 74 of file def_struct.h.

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

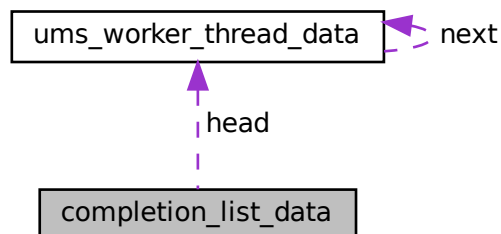
- Headers/ **def_struct.h**

4.3 completion_list_data Struct Reference

Struct used to share **completion_list_data** (p. ??) info with the kernel.

```
#include <user_struct.h>
```

Collaboration diagram for completion_list_data:



Data Fields

- ums_pid_t **id**
- struct **ums_worker_thread_data** * **head**
- int **requested_size**

4.3.1 Detailed Description

Struct used to share **completion_list_data** (p. ??) info with the kernel.

Definition at line 27 of file user_struct.h.

4.3.2 Field Documentation

4.3.2.1 head

```
struct ums_worker_thread_data* head
```

pointer to an array of **ums_worker_thread_data** (p. ??)

Definition at line 30 of file user_struct.h.

4.3.2.2 id

```
ums_pid_t id
```

identifier for the completion list

Definition at line 29 of file user_struct.h.

4.3.2.3 requested_size

```
int requested_size
```

size requested from the kernel to be allocated in user space for the DEQUEUE

Definition at line 31 of file user_struct.h.

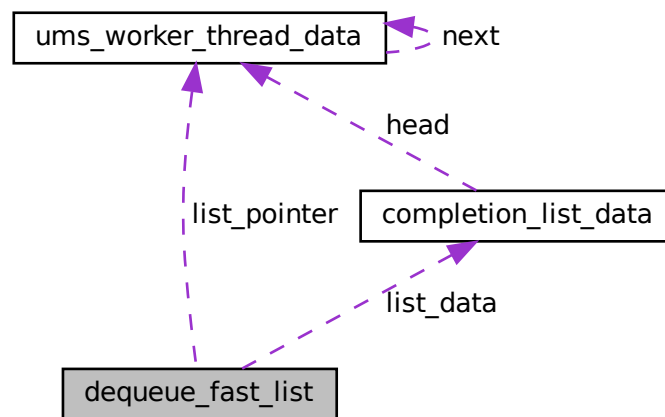
The documentation for this struct was generated from the following file:

- Headers/ **user_struct.h**

4.4 dequeue_fast_list Struct Reference

Struct used by a UMS Scheduler thread to maintain a pre-allocated completion_list_data* used in the dequeue function.

Collaboration diagram for dequeue_fast_list:



Data Fields

- struct `completion_list_data *` `list_data`
- int `cur_size`
- int `origin_size`
- struct `ums_worker_thread_data **` `list_pointer`

4.4.1 Detailed Description

Struct used by a UMS Scheduler thread to maintain a pre-allocated completion_list_data* used in the dequeue function.

Definition at line 13 of file ums_user_library.c.

4.4.2 Field Documentation

4.4.2.1 cur_size

```
int cur_size
```

cur size of list_data

Definition at line 17 of file ums_user_library.c.

4.4.2.2 list_data

```
struct completion_list_data* list_data
```

pointer to the **completion_list_data** (p. ??) that will be passed to the LKM

Definition at line 16 of file ums_user_library.c.

4.4.2.3 list_pointer

```
struct ums_worker_thread_data** list_pointer
```

array that contains all the pointer used in list_data, this array is used only to free the element, since the kernel will overwrite ums_worker_thread_data->next in case of "NULL"

Definition at line 19 of file ums_user_library.c.

4.4.2.4 origin_size

```
int origin_size
```

origin size of list_data

Definition at line 18 of file ums_user_library.c.

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

- Headers/ums_user_library.c

4.5 sched_thread_data Struct Reference

Struct used to share **sched_thread_data** (p. ??) info with the kernel.

```
#include <user_struct.h>
```

Data Fields

- unsigned long **completion_list**
- int(* **entry_point**)(void *)
- void * **args**

4.5.1 Detailed Description

Struct used to share **sched_thread_data** (p. ??) info with the kernel.

Definition at line 16 of file user_struct.h.

4.5.2 Field Documentation

4.5.2.1 args

```
void*  args
```

pointer to the entry_point params(usually you need to pass at least the file descriptor)

Definition at line 19 of file user_struct.h.

4.5.2.2 completion_list

```
unsigned long  completion_list
```

ID of the **completion_list** (p. ??)

Definition at line 17 of file user_struct.h.

4.5.2.3 entry_point

```
int (* entry_point(void *))
```

pointer to the scheduling function

Definition at line 18 of file user_struct.h.

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

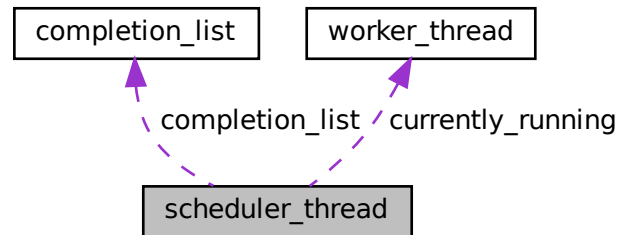
- Headers/ **user_struct.h**

4.6 scheduler_thread Struct Reference

Structure used by an UMS Scheduler to maintain scheduler thread infos, used in ENTER_SCHEDULING_MODE.

```
#include <def_struct.h>
```

Collaboration diagram for scheduler_thread:



Data Fields

- pid_t **pid**
- int(* **entry_point**)(void *)
- void * **args**
- int **yield_priority**
- int **nesting**
- struct **worker_thread** * **currently_running**
- struct list_head **worker_lifo**
- struct list_head **next**
- struct **completion_list** * **completion_list**
- spinlock_t **sched_lock**
- int **destroy**
- int **started**
- int **ended**
- int **numb_switch**
- struct proc_dir_entry * **ent**
- struct proc_dir_entry * **ent_write**
- struct proc_dir_entry * **father**
- time64_t **last_switch_time**
- time64_t **mean_switch_time**

4.6.1 Detailed Description

Structure used by an UMS Scheduler to maintain scheduler thread infos, used in ENTER_SCHEDULING_MODE.

Definition at line 95 of file def_struct.h.

4.6.2 Field Documentation

4.6.2.1 args

`void* args`

scheduling function

Definition at line 99 of file def_struct.h.

4.6.2.2 currently_running

`struct worker_thread* currently_running`

pointer to the currently running worker

Definition at line 102 of file def_struct.h.

4.6.2.3 destroy

`int destroy`

flag used for determining when the thread has to exit after a ums_release(fd)

Definition at line 108 of file def_struct.h.

4.6.2.4 ended

`int ended`

counter of the worker thread that has completed

Definition at line 110 of file def_struct.h.

4.6.2.5 ent

`struct proc_dir_entry* ent`

pointer to proc_entry for "/proc/ums/scheduler_id/schedulers/pid"

Definition at line 112 of file def_struct.h.

4.6.2.6 ent_write

```
struct proc_dir_entry* ent_write
```

pointer to proc_entry for "/proc/ums/scheduler_id/schedulers/pid/workers"

Definition at line 113 of file def_struct.h.

4.6.2.7 entry_point

```
int (* entry_point(void *))
```

scheduling function

Definition at line 98 of file def_struct.h.

4.6.2.8 father

```
struct proc_dir_entry* father
```

pointer to proc_entry for "/proc/ums/scheduler_id/schdulers"

Definition at line 114 of file def_struct.h.

4.6.2.9 last_switch_time

```
time64_t last_switch_time
```

time elapsed to complete the last context switch

Definition at line 115 of file def_struct.h.

4.6.2.10 mean_switch_time

```
time64_t mean_switch_time
```

time elapsed to complete the last context switch

Definition at line 116 of file def_struct.h.

4.6.2.11 nesting

```
int nesting
```

used with completion_list->sem_counter to take priority on the **completion_list** (p. ??) after a yield

Definition at line 101 of file def_struct.h.

4.6.2.12 numb_switch

```
int numb_switch
```

counter of the worker thread that switched

Definition at line 111 of file def_struct.h.

4.6.2.13 pid

```
pid_t pid
```

PID of the original pthread

Definition at line 97 of file def_struct.h.

4.6.2.14 started

```
int started
```

counter of the worker thread which it has started executing

Definition at line 109 of file def_struct.h.

4.6.2.15 yield_priority

```
int yield_priority
```

flag used for determining if the current scheduler thread has already done a down(&sem)

Definition at line 100 of file def_struct.h.

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

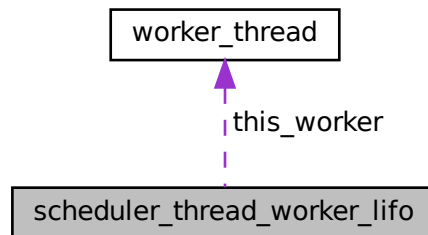
- Headers/ **def_struct.h**

4.7 scheduler_thread_worker_lifo Struct Reference

Strucute used by a UMS Scheduler thread to save the order of the worker threads called.

```
#include <def_struct.h>
```

Collaboration diagram for scheduler_thread_worker_lifo:



Data Fields

- struct **worker_thread** * **this_worker**
- struct list_head **next**

4.7.1 Detailed Description

Strucute used by a UMS Scheduler thread to save the order of the worker threads called.

Definition at line 83 of file def_struct.h.

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

- Headers/ **def_struct.h**

4.8 ums_device_data_s Struct Reference

structure containing the data needed by module to initialize the UMS device driver

```
#include <def_struct.h>
```

Data Fields

- struct class * **class**
- struct device * **device**
- int **major**

4.8.1 Detailed Description

structure containing the data needed by module to initialize the UMS device driver

Definition at line 34 of file `def_struct.h`.

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

- Headers/ `def_struct.h`

4.9 UMS_sched_data Struct Reference

Structure used by the kernel module to maintain different UMS Scheduler.

```
#include <def_struct.h>
```

Data Fields

- struct list_head **scheduler_thread**
- struct list_head **next**
- unsigned long **owner**
- unsigned long int **id_counter**
- struct proc_dir_entry * **ent**
- struct proc_dir_entry * **ent_write**
- struct proc_dir_entry * **father**

4.9.1 Detailed Description

Structure used by the kernel module to maintain different UMS Scheduler.

Definition at line 126 of file `def_struct.h`.

4.9.2 Field Documentation

4.9.2.1 ent

```
struct proc_dir_entry* ent
```

pointer to proc_entry for "/proc/ums/id_counter"

Definition at line 134 of file `def_struct.h`.

4.9.2.2 ent_write

```
struct proc_dir_entry* ent_write
```

pointer to proc_entry for "/proc/ums/id_counter/schedulers"

Definition at line 135 of file def_struct.h.

4.9.2.3 father

```
struct proc_dir_entry* father
```

pointer to proc_entry for "/proc/ums"

Definition at line 136 of file def_struct.h.

4.9.2.4 id_counter

```
unsigned long int id_counter
```

ID of the UMS Scheduler(we are using filep)

Definition at line 133 of file def_struct.h.

4.9.2.5 owner

```
unsigned long owner
```

pid of the owner

Definition at line 132 of file def_struct.h.

4.9.2.6 scheduler_thread

```
struct list_head scheduler_thread
```

used to maintain al the UMS Scheduler threads related to this UMS Scheduler

Definition at line 129 of file def_struct.h.

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

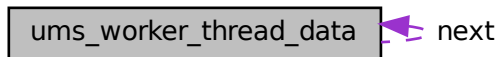
- Headers/ **def_struct.h**

4.10 ums_worker_thread_data Struct Reference

Struct used to share **ums_worker_thread_data** (p. ??) infos with the kernel.

```
#include <user_struct.h>
```

Collaboration diagram for ums_worker_thread_data:



Data Fields

- ums_pid_t **pid**
- ums_pid_t **completion_list_id**
- int(* **work**)(void *)
- void * **args**
- struct **ums_worker_thread_data** * **next**

4.10.1 Detailed Description

Struct used to share **ums_worker_thread_data** (p. ??) infos with the kernel.

Definition at line 42 of file user_struct.h.

4.10.2 Field Documentation

4.10.2.1 args

```
void* args
```

argument that will be passed to **work()** (p. ??)

Definition at line 47 of file user_struct.h.

4.10.2.2 completion_list_id

```
ums_pid_t completion_list_id
```

unsigned long identifier for the completion list who own this worker

Definition at line 45 of file user_struct.h.

4.10.2.3 next

```
struct ums_worker_thread_data* next
```

pointer to the next worker (this variable is !null onli if retrived from a completion_list_data->head)

Definition at line 48 of file user_struct.h.

4.10.2.4 pid

```
ums_pid_t pid
```

identifier of the worker thread

Definition at line 44 of file user_struct.h.

4.10.2.5 work

```
int (* work(void *))
```

pointer to a function that will be executed by the worker thread

Definition at line 46 of file user_struct.h.

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

- Headers/ **user_struct.h**

4.11 worker_thread Struct Reference

Strucute used by a UMS Scheduler to mantain Worker Thread info, used in ADD_WORKER.

```
#include <def_struct.h>
```

Data Fields

- int **state**
- ums_pid_t **pid**
- unsigned long **completion_list_id**
- spinlock_t **execute_lock**
- struct list_head **next**
- struct task_struct * **task_struct**
- int(* **work**)(void *)
- void * **args**
- unsigned long **numb_switch**
- time64_t **running_time**
- time64_t **starting_time**

4.11.1 Detailed Description

Strucute used by a UMS Scheduler to maintain Worker Thread info, used in ADD_WORKER.

Definition at line 46 of file def_struct.h.

4.11.2 Field Documentation

4.11.2.1 args

void* **args**

arguments for the work function

Definition at line 55 of file def_struct.h.

4.11.2.2 completion_list_id

unsigned long **completion_list_id**

id of the completion list who own this this worker

Definition at line 50 of file def_struct.h.

4.11.2.3 execute_lock

spinlock_t **execute_lock**

used for concurrency while managing worker

Definition at line 51 of file def_struct.h.

4.11.2.4 numb_switch

```
unsigned long numb_switch
```

args that will be passed to the **work()** (p. ??) function

Definition at line 56 of file def_struct.h.

4.11.2.5 pid

```
ums_pid_t pid
```

id of the worker

Definition at line 49 of file def_struct.h.

4.11.2.6 state

```
int state
```

State of the worker

Definition at line 48 of file def_struct.h.

4.11.2.7 task_struct

```
struct task_struct* task_struct
```

task struct related to this worker

Definition at line 53 of file def_struct.h.

4.11.2.8 work

```
int (* work(void *))
```

function to be executed from this worker

Definition at line 54 of file def_struct.h.

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

- Headers/ **def_struct.h**

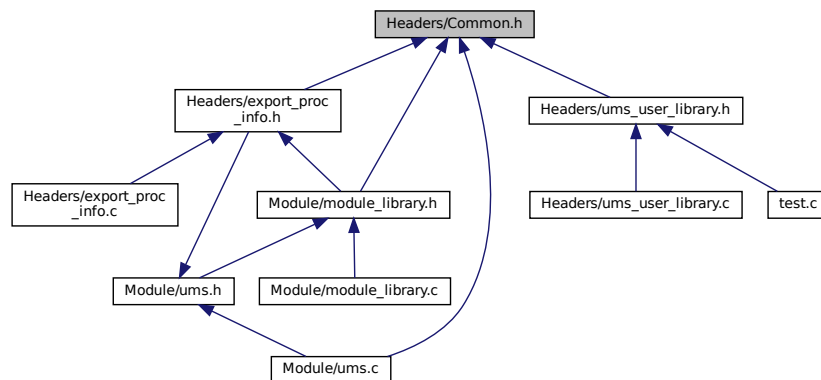
Chapter 5

File Documentation

5.1 Headers/Common.h File Reference

Contains constants and macros used both by the module and the user library.

This graph shows which files directly or indirectly include this file:



Macros

- `#define DEVICE_NAME "ums_scheduler"`
- `#define MODULE_NAME_LOG "UMS_SCHEDULER: "`
- `#define CLASS_NAME "ums_scheduler_class"`
- `#define OPEN_PATH "/dev/ums_scheduler"`
- `#define DEBUG_M_VAR 0`
- `#define DEBUG_DO_PPRINTK(fmt, ...)`
function used by the kernel to print debug msgs
- `#define DEBUG_DO_PRINTK(fmt)`
function used by the kernel to print debug msgs
- `#define ERR_PPRINTK(fmt, ...) printk(KERN_ERR fmt, ##__VA_ARGS__);`
function used by the kernel to print error msgs

- **#define ERR_PRINTK**(fmt) printk(KERN_ERR fmt);
function used by the kernel to print error msgs
- **#define NOTIFY_EXEC_ERR_M_VAR** 0
- **#define NOTIFY_EXEC_ERR_PPRINTK**(fmt, ...) if(NOTIFY_EXEC_ERR_M_VAR)printk(KERN_ERR fmt, ##__VA_ARGS__);
function used by the kernel to print error_exec msgs
- **#define MAGIC_K_VALUE** 'K'
- **#define ENTER_UMS_SCHEDULING_MODE** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 0, sizeof(sched_thread_data))
ENTER_UMS_SCHEDULING_MODE.
- **#define DEQUEUE_UMS_COMPLETION_LIST_ITEMS** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 1, 8UL)
DEQUEUE_UMS_COMPLETION_LIST_ITEMS.
- **#define EXECUTE_UMS_THREAD** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 2, 8UL)
EXECUTE_UMS_THREAD.
- **#define UMS_THREAD_YIELD** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 3, 8UL)
UMS_THREAD_YIELD.
- **#define EXIT_WORKER_THREAD** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 4, 8UL)
EXIT_WORKER_THREAD.
- **#define CREATE_COMP_LIST** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 5, 8UL)
CREATE_COMP_LIST.
- **#define ADD_WORKER** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 6, 8UL)
ADD_WORKER.
- **#define DESTROY_COMP_LIST** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 7, 8UL)
DESTROY_COMP_LIST.
- **#define RELEASE_UMS** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 8, 8UL)
RELEASE_UMS.
- **#define DEQUEUE_SIZE_REQUEST** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 9, 8UL)
DEQUEUE_SIZE_REQUEST.
- **#define EXIT_FROM_YIELD** _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 10, 8UL)
EXIT_FROM_YIELD.

5.1.1 Detailed Description

Contains constants and macros used both by the module and the user library.

5.1.2 Macro Definition Documentation

5.1.2.1 ADD_WORKER

```
#define ADD_WORKER _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 6, 8UL)
```

ADD_WORKER.

Parameters

<i>ums_worker_thread_data*</i>	pointer to ums_worker_thread_data (p. ??)
--------------------------------	--------------------------------------------------

Returns

0 on success, err otherwise and errno is setted.

This function takes as input **ums_worker_thread_data*

add the *ums_worker_thread_data* to the desired completion list

return 0 on success, err otherwise and errno is setted.

in the case of readers/writers or publish/subscribe or master/slaves use 2 different completion_list

errno:

-EACCES, the completion list is marked to be destroyed.

Definition at line 230 of file Common.h.

5.1.2.2 CREATE_COMP_LIST

```
#define CREATE_COMP_LIST _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 5, 8UL)
```

CREATE_COMP_LIST.

Parameters

<i>completion_list_data*</i>	pointer to completion_list_data (p. ??)
------------------------------	------------------------------------------------

Returns

0 on success, err otherwise and errno is setted

this function takes as input **completion_list_data*

generate a new **completion_list** (p. ??), and store the id in the passed **completion_list_data*

return 0 on success, err otherwise and errno is setted

errno: -ENOMEM no more space in the kernel

Definition at line 206 of file Common.h.

5.1.2.3 DEBUG_DO_PPRINTK

```
#define DEBUG_DO_PPRINTK(  
    fmt,  
    ... )
```

Value:

```
if (DEBUG_M_VAR)  
{  
    printk(KERN_INFO fmt, ##__VA_ARGS__); \
```

function used by the kernel to print debug msgs

Definition at line 26 of file Common.h.

5.1.2.4 DEBUG_DO_PRINTK

```
#define DEBUG_DO_PRINTK(  
    fmt )
```

Value:

```
if (DEBUG_M_VAR)  
{  
    printk(KERN_INFO fmt); \
```

function used by the kernel to print debug msgs

Definition at line 36 of file Common.h.

5.1.2.5 DEQUEUE_SIZE_REQUEST

```
#define DEQUEUE_SIZE_REQUEST _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 9, 8UL)
```

DEQUEUE_SIZE_REQUEST.

Parameters

<i>pointer</i>	to completion_list_data (p. ??)
----------------	---------------------------------

Returns

0 on success, err otherwise and errno is setted

This function return the size to be allocated in user space
in order to recive the list of ums_worker_thread_data(user space struct)

copy to args *completion_list_data with:
-requested_size setted

Definition at line 275 of file Common.h.

5.1.2.6 DEQUEUE_UMS_COMPLETION_LIST_ITEMS

```
#define DEQUEUE_UMS_COMPLETION_LIST_ITEMS _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 1, 8UL)
```

DEQUEUE_UMS_COMPLETION_LIST_ITEMS.

Parameters

<i>completion_list_data*</i>	pointer to completion_list_data (p. ??)
------------------------------	------------------------------------------------

Returns

0 on success, err otherwise and errno is setted

This function read from args *completion_list_data

if the completion list is empty(no worker to be executed) the UMS scheduler thread will sleep if the completion list is not empty: 1)the functio will check the max space allowed to be written from completion_list_Data 2)then copy (only the allowed infos) **worker_thread(kernel structre)** (p. ??) to each **ums_worker_thread_data(user structure)** (p. ??) entry in **completion_list_data** (p. ??) if no more space or no more worker the function exits return 0 on succedd, err otherwise errno wil be set errno:

- 404: the UMS called the UMS_RELEASE, this pthread should stop calling the device we suggest to return 0(see **ums_user_library.c** (p. ??) example)

Definition at line 109 of file Common.h.

5.1.2.7 DESTROY_COMP_LIST

```
#define DESTROY_COMP_LIST _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 7, 8UL)
```

DESTROY_COMP_LIST.

Parameters

<i>completion_list_data*</i>	pointer to a completion_list_data (p. ??)
------------------------------	--------------------------------------------------

Returns

0 on success, err otherwise and errno is setted.

```
DESTROY_COMP_LIST
takes as input a *completion_list_data
this function simply mark the completion_list to destroy
```

Definition at line 245 of file Common.h.

5.1.2.8 ENTER_UMS_SCHEDULING_MODE

```
#define ENTER_UMS_SCHEDULING_MODE _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 0, sizeof( sched↔  
_thread_data))
```

ENTER_UMS_SCHEDULING_MODE.

Parameters

<i>sched_thread_data*</i>	pointer to sched_thread_data (p. ??)
---------------------------	---------------------------------------------

Returns

0 on success, err otherwise and errno is setted

```
reads from args *sched_thread_data with:  
-entrypoint  
-args(for the entry point)  
-the completion list id
```

Then call create_ums_thread.

returns 0 on success, err otherwise, the errno is setted

Definition at line 84 of file Common.h.

5.1.2.9 ERR_PPRINTK

```
#define ERR_PPRINTK(  
    fmt,  
    ... ) printk(KERN_ERR fmt, ##__VA_ARGS__);
```

function used by the kernel to print error msgs

Definition at line 46 of file Common.h.

5.1.2.10 ERR_PRINTK

```
#define ERR_PRINTK(  
    fmt ) printk(KERN_ERR fmt);
```

function used by the kernel to print error msgs

Definition at line 51 of file Common.h.

5.1.2.11 EXECUTE_UMS_THREAD

```
#define EXECUTE_UMS_THREAD _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 2, 8UL)
```

EXECUTE_UMS_THREAD.

Parameters

<code>ums_worker_thread_data*</code>	pointer to <code>ums_worker_thread_data</code> (p. ??)
--------------------------------------	--------------------------------------------------------

Returns

0 on success, err otherwise and errno is setted

called from a scheduler thread, it executes the passed worker thread by switching the entire context

this function takes as input a `*ums_worker_thread_data`
 if the worker is not `RUNNABLE` or not exists the function exits
 otherwise set all the information in the current scheduler and then return `ums_worker_thread_data` filled
 with at least `*work` and `*args`
 The worker will be set to `RUNNING`

errno:
 - `EFBIG`, no more space in kernel to allow a new `scheduler_thread_worker_lifo`
 - `EBADF`, worker not found
 - `EACCES`, the worker is not `RUNNABLE`
 with one of this errno the default operation is to try to execute the next worker;

Definition at line 136 of file `Common.h`.

5.1.2.12 EXIT_FROM_YIELD

```
#define EXIT_FROM_YIELD _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 10, 8UL)
```

`EXIT_FROM_YIELD`.

Returns

0 on success, err otherwise and errno is setted

This is a helper function, should be called after an `UMS_THREAD_YIELD`
 this simply decrement a yield counter in the scheduler
 not calling this function will have no consequences on the execution
 but scheduler infos will be messed up, so that `RELEASE` will fails when call

Definition at line 288 of file `Common.h`.

5.1.2.13 EXIT_WORKER_THREAD

```
#define EXIT_WORKER_THREAD _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 4, 8UL)
```

`EXIT_WORKER_THREAD`.

Parameters

<code>sched_thread_data*</code>	pointer to <code>sched_thread_data</code> (p. ??)
---------------------------------	---------------------------------------------------

Returns

0 on success, err otherwise and errno is setted

This functions must be called at the end of a worker_thread *work function.
This function takes as input *sched_thread_data

Then the worker will be set to EXIT, and rmeoved from:
the completion_list and from the list of the UMS scheduler thread.

at the end like the execute pass sched_thread_data to the user filled with entry_point and args

errno:
- EBUSY, no other worker to execute, but you come from a UMS_THREAD_YIELD,
you have simply to return 0 from the current function;

Definition at line 184 of file Common.h.

5.1.2.14 NOTIFY_EXEC_ERR_PPRINTK

```
#define NOTIFY_EXEC_ERR_PPRINTK(  
    fmt,  
    ... ) if(NOTIFY_EXEC_ERR_M_VAR)printk(KERN_ERR fmt, ##__VA_ARGS__);
```

function used by the kernel to print error_exec msgs

Definition at line 58 of file Common.h.

5.1.2.15 RELEASE_UMS

```
#define RELEASE_UMS _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 8, 8UL)
```

RELEASE_UMS.

Returns

0 on success, err otherwise and errno is setted. RELEASE_UMS

this funciton simply relelase the current UMS Scheduler

Definition at line 259 of file Common.h.

5.1.2.16 UMS_THREAD_YIELD

```
#define UMS_THREAD_YIELD _IOC(_IOC_WRITE | _IOC_READ, MAGIC_K_VALUE, 3, 8UL)
```

UMS_THREAD_YIELD.

Parameters

<code>sched_thread_data*</code>	pointer to <code>sched_thread_data</code> (p. ??)
---------------------------------	---------------------------------------------------

Returns

err and errno is setted

UMS_THREAD_YIELD

called from a worker thread, it pauses the execution of the current thread and the UMS scheduler entry point

Takes as input `*sched_thread_data` and fill it with `*entry_point` and `*args`
The worker will be set to YIELD

* errno:

- EBUSY, no other workers, this mean that you should just call the EXIT_FROM_YIELD
- EAGAIN, worker found, you have to call the entypoint before call the EXIT_FROM_YIELD

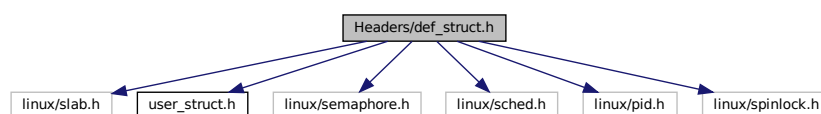
Definition at line 159 of file Common.h.

5.2 Headers/def_struct.h File Reference

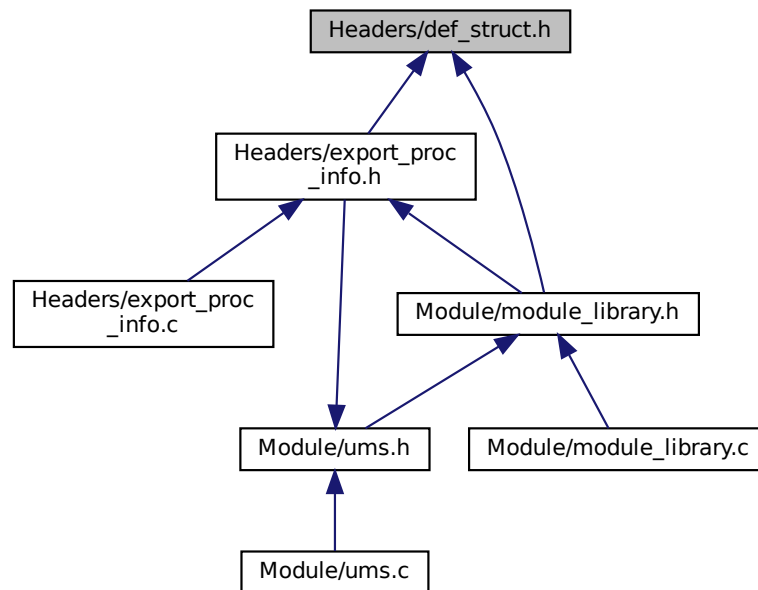
Contains the definitions of every custom type used in the Kernel module code.

```
#include <linux/slab.h>
#include "user_struct.h"
#include <linux/semaphore.h>
#include <linux/sched.h>
#include <linux/pid.h>
#include <linux/spinlock.h>
```

Include dependency graph for def_struct.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **ums_device_data_s**
structure containing the data needed by module to initialize the UMS device driver
- struct **worker_thread**
Structure used by a UMS Scheduler to maintain Worker Thread info, used in ADD_WORKER.
- struct **completion_list**
Structure used by an UMS Scheduler to maintain Completion list infos, used in CREATE_COMP_LIST.
- struct **scheduler_thread_worker_lifo**
Structure used by a UMS Scheduler thread to save the order of the worker threads called.
- struct **scheduler_thread**
Structure used by an UMS Scheduler to maintain scheduler thread infos, used in ENTER_SCHEDULING_MODE.
- struct **UMS_sched_data**
Structure used by the kernel module to maintain different UMS Scheduler.

Macros

- #define **EXIT** 0
- #define **RUNNING** 1
- #define **RUNNABLE** 2
- #define **IOWAIT** 3
- #define **SYSWAIT** 4
- #define **YIELD** 5
- #define **UNITIALIZED** 6
- #define **PRIO** 1
- #define **NOT_PRIO** 0

Typedefs

- typedef struct **ums_device_data_s** **ums_device_data_t**
structure containing the data needed by module to initialize the UMS device driver
- typedef struct **worker_thread** **worker_thread**
Strucute used by a UMS Scheduler to mantain Worker Thread info, used in ADD_WORKER.
- typedef struct **completion_list** **completion_list**
Structure used by an UMS Scheduler to mantain Completion list infos, used in CREATE_COMP_LIST.
- typedef struct **scheduler_thread_worker_lifo** **scheduler_thread_worker_lifo**
Strucute used by a UMS Scheduler thread to save the order of the worker threads called.
- typedef struct **scheduler_thread** **scheduler_thread**
Structure used by an UMS Scheduler to mantain scheduler thread infos, used in ENTER_SCHEDULING_MODE.
- typedef struct **UMS_sched_data** **UMS_sched_data**
Structure used by the kernel module to mantain differrent UMS Scheduler.

5.2.1 Detailed Description

Contains the definitions of every custom type used in the Kernel module code.

5.2.2 Typedef Documentation

5.2.2.1 completion_list

```
typedef struct completion_list completion_list
```

Structure used by an UMS Scheduler to mantain Completion list infos, used in CREATE_COMP_LIST.

5.2.2.2 scheduler_thread

```
typedef struct scheduler_thread scheduler_thread
```

Structure used by an UMS Scheduler to mantain scheduler thread infos, used in ENTER_SCHEDULING_MODE.

5.2.2.3 scheduler_thread_worker_lifo

```
typedef struct scheduler_thread_worker_lifo scheduler_thread_worker_lifo
```

Strucute used by a UMS Scheduler thread to save the order of the worker threads called.

5.2.2.4 UMS_sched_data

```
typedef struct UMS_sched_data UMS_sched_data
```

Structure used by the kernel module to maintain different UMS Scheduler.

5.2.2.5 worker_thread

```
typedef struct worker_thread worker_thread
```

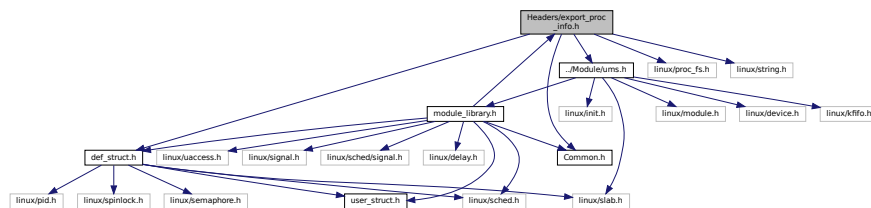
Strucute used by a UMS Scheduler to maintain Worker Thread info, used in ADD_WORKER.

5.3 Headers/export_proc_info.h File Reference

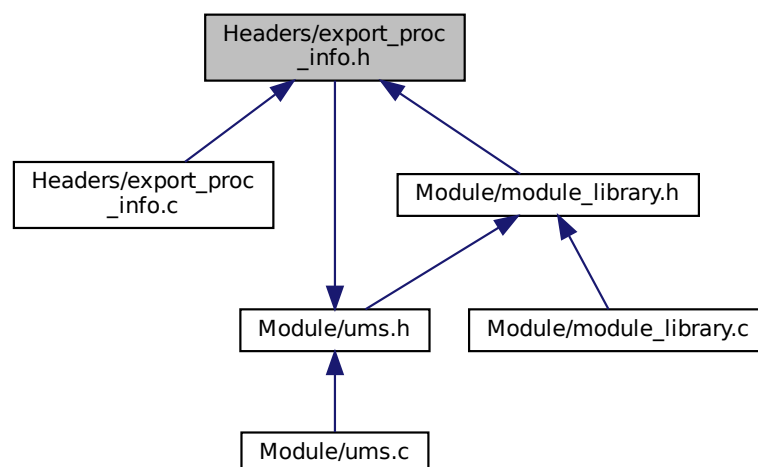
This file contains, macros and functions used to manage the /proc files.

```
#include "def_struct.h"
#include <linux/proc_fs.h>
#include "Common.h"
#include "../Module/ums.h"
#include <linux/string.h>
```

Include dependency graph for export_proc_info.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define ums_standard_path "ums"`
- `#define schedulers_dir "schedulers"`
- `#define workers_dir "workers"`
- `#define scheduler_info "info"`
- `#define scheduler_standard_path "ums/%lu"`
- `#define scheduler_thread_standard_path "ums/%lu/scheduler/%lu"`
- `#define worker_standard_path "ums/%lu/scheduler/%lu/workers/%lu"`
- `#define PATH_CREATE_SIZE 1024`
- `#define PATH_RM_SIZE 2048`
- `#define RETRIVE_NAME_SIZE 2048`
- `#define WORKER 1`
- `#define SCHEDULER 2`
- `#define SCHEDULER_THREAD 3`
- `#define BUFSIZE 4096`

Functions

- `int create_proc_ums (struct proc_dir_entry **ent)`
Create an ums dir in /proc.
- `int create_proc_ums_scheduler (unsigned long filep, struct UMS_sched_data *scheduler, struct proc_dir_entry *father)`
Create a <filep> dir under /proc/ums, and /proc/ums/<filep>/shedulers, <filep> is the identifier of an UMS Scheduler.
- `int create_proc_ums_scheduler_thread (unsigned long filep, struct scheduler_thread *sched, struct proc_dir_entry *father)`
Create: a <pid> dir under /proc/ums/<filep>/schedulers, file "info" and a "workers" dir under /proc/ums/<filep>/schedulers/<pid>. <pid> is the identifier of an UMS Scheduler thread.
- `int create_proc_worker (struct worker_thread *worker, struct scheduler_thread *sched)`
Create a <pid> file under /proc/ums/<filep>/schedulers/<father_pid>/workers.
- `int rm_proc_ums (void)`
Remove the "ums" dir under /proc.
- `int rm_proc_ums_scheduler (unsigned long filep, struct UMS_sched_data *scheduler)`
remove the <filep> dir under /proc/ums
- `int rm_proc_ums_scheduler_thread (unsigned long filep, struct scheduler_thread *sched)`
remove the <pid> dir under /proc/ums/<filep>/schedulers
- `int rm_proc_worker (struct worker_thread *worker, struct scheduler_thread *sched)`
remove the <pid> file under /proc/ums/<filep>/schedulers/<father-pid>/workers

5.3.1 Detailed Description

This file contains, macros and functions used to manage the /proc files.

5.3.2 Function Documentation

5.3.2.1 create_proc_ums()

```
int create_proc_ums (
    struct proc_dir_entry ** ent )
```

Create an ums dir in /proc.

Parameters

<i>ent</i>	double pointer to a <code>proc_dir_entry</code> that will receive the new pointer to the dir
------------	----------------------------------------------------------------------------------------------

Returns

int 0 on success, -1 otherwise

Definition at line 101 of file `export_proc_info.c`.

5.3.2.2 create_proc_ums_scheduler()

```
int create_proc_ums_scheduler (
    unsigned long filep,
    struct UMS_sched_data * scheduler,
    struct proc_dir_entry * father )
```

Create a `<filep>` dir under `/proc/ums`, and `/proc/ums/<filep>/shedulers`, `<filep>` is the identifier of an UMS Scheduler.

Parameters

<i>filep</i>	id of the UMS Scheduler
<i>scheduler</i>	pointer to the <code>UMS_Sched_data</code> that will receive in the <code>[ent,ent_write,father]</code> the <code>proc_dir_entry</code> pointers.
<i>father</i>	pointer to the father <code>proc_dir_entry</code>

Returns

int 0 on success, err otherwise

Definition at line 111 of file `export_proc_info.c`.

5.3.2.3 create_proc_ums_scheduler_thread()

```
int create_proc_ums_scheduler_thread (
    unsigned long filep,
    struct scheduler_thread * sched,
    struct proc_dir_entry * father )
```

Create: a `<pid>` dir under `/proc/ums/<filep>/shedulers`, file "info" and a "workers" dir under `/proc/ums/<filep>/shedulers/<pid>`. `<pid>` is the identifier of an UMS Scheduler thread.

Parameters

<i>filep</i>	id of the UMS Scheduler
<i>sched</i>	pointer to the scheduler_thread (p. ??) that will receive in the <code>[ent,ent_write,father]</code> the <code>proc_dir_entry</code> pointers.
<i>father</i>	pointer to the father <code>proc_dir_entry</code>

Returns

int 0 on success, err otherwise

Definition at line 140 of file export_proc_info.c.

5.3.2.4 create_proc_worker()

```
int create_proc_worker (
    struct worker_thread * worker,
    struct scheduler_thread * sched )
```

Create a <pid> file under /proc/ums/<filep>/schedulers/<father_pid>/workers.

Parameters

<i>worker</i>	pointer to a worker_thread (p. ??) struct to retrieve the worker infos
<i>sched</i>	pointer to the scheduler thread who manage this worker

Returns

int 0 on success, err otherwise

Definition at line 168 of file export_proc_info.c.

5.3.2.5 rm_proc_ums()

```
int rm_proc_ums (
    void )
```

Remove the "ums" dir under /proc.

Returns

int 0 on success, err otherwise

Definition at line 183 of file export_proc_info.c.

5.3.2.6 rm_proc_ums_scheduler()

```
int rm_proc_ums_scheduler (
    unsigned long filep,
    struct UMS_sched_data * scheduler )
```

remove the <filep> dir under /proc/ums

Parameters

<i>filep</i>	id of the UMS Scheduler
<i>scheduler</i>	pointer to the UMS Scheduler

Returns

int 0 on success, err otherwise

Definition at line 188 of file export_proc_info.c.

5.3.2.7 rm_proc_ums_scheduler_thread()

```
int rm_proc_ums_scheduler_thread (
    unsigned long filep,
    struct scheduler_thread * sched )
```

remove the <pid> dir under /proc/ums/<filep>/schedulers

Parameters

<i>filep</i>	id of the UMS Scheduler who manage this Ums scheduler thread
<i>sched</i>	pointer to the UMS scheduler thread

Returns

int 0 on success, err otherwise

Definition at line 201 of file export_proc_info.c.

5.3.2.8 rm_proc_worker()

```
int rm_proc_worker (
    struct worker_thread * worker,
    struct scheduler_thread * sched )
```

remove the <pid> file under /proc/ums/<filep>/schedulers/<father-pid>/workers

Parameters

<i>worker</i>	pointer to the desired worker to eliminate
<i>sched</i>	pointer to the UMS scheduler thread who manage this worker

Returns

int 0 on success, err otherwise

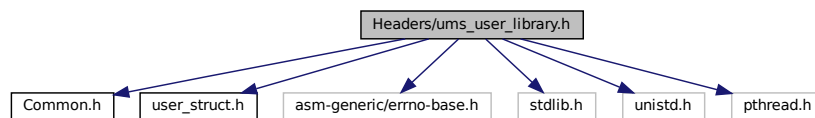
Definition at line 213 of file export_proc_info.c.

5.4 Headers/ums_user_library.h File Reference

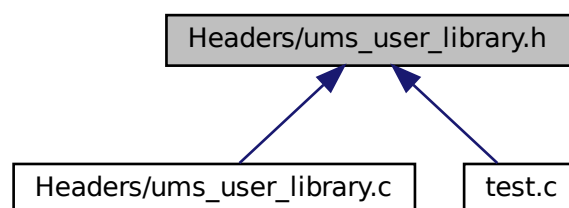
This library contains the definitions of every function that can be used from user space to interact with the UMS Scheduler device.

```
#include "Common.h"
#include "user_struct.h"
#include <asm-generic/errno-base.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
```

Include dependency graph for ums_user_library.h:



This graph shows which files directly or indirectly include this file:

**Macros**

- #define **DEBUG_USER_M_VAR** 0
- #define **DEBUG_USER_DO_PPRINTF**(fmt, ...)
 - function used by the library to print debug msgs*
- #define **DEBUG_USER_DO_PRINTF**(fmt)
 - function used by the library to print debug msgs*

- `#define ERR_USER_M_VAR 1`
- `#define ERR_USER_PPRINTF(fmt, ...)`
function used by the library to print err msgs
- `#define ERR_USER_PRINTF(fmt)`
function used by the library to print err msgs
- `#define ERR_USER_EXEC_M_VAR 0`
- `#define ERR_USER_EXEC_PPRINTF(fmt, ...)`
function used by the library to print err_exec msgs
- `#define ERR_USER_EXEC_PRINTF(fmt)`
function used by the library to print err_exec msgs

Functions

- `int ums_init ()`
Initialize a new UMS.
- `int ums_close (int fd)`
Release all the structures related to the fd UMS.
- `int enter_ums_scheduling_mode (int fd, int(*entry_point)(void *), void *entrypoint_args, ums_pid_t completion_list_id)`
Converts a standard pthread in a UMS Scheduler thread, the function takes as input a file descriptor, a completion list ID of worker threads and a entry point function.
- `completion_list_data * dequeue_ums_completion_list_items (int fd)`
called from the scheduler thread obtains a list of current available thread to be run, if no thread is available to be run the function should be blocking until a thread becomes available
- `int execute_ums_thread (int fd, ums_pid_t worker_id, completion_list_data *list_data)`
called from a scheduler thread, it executes the passed worker thread by switching the entire context, if -EACCES or -EBADF are returned try again with a different worker_id.
- `int ums_thread_yield (int fd)`
called from a worker thread, it pauses the execution of the current thread and the UMS scheduler entry point is executed for determining the next thread to be scheduled;
- `int exit_worker_thread (int fd)`
This function must be called at the end of every worker_thread (p. ??).
- `completion_list_data * create_completion_list (int fd)`
Create a completion list object, the ID is located in create_completion_list->id, in the case of readers/writers or publish/subscribe or master/slaves use 2 different completion_list (p. ??) for the two groups of workers.
- `int add_worker_thread (int fd, int(*work)(void *), void * args, ums_pid_t completion_list_id)`
Create and add a worker thread to the desired completion list.
- `int destroy_comp_list (int fd, ums_pid_t completion_list_id)`
destroy the selected completion list
- `int release_ums (int fd)`
destroy the UMS

5.4.1 Detailed Description

This library contains the definitions of every function that can be used from user space to interact with the UMS Scheduler device.

5.4.2 Macro Definition Documentation

5.4.2.1 DEBUG_USER_DO_PPRINTF

```
#define DEBUG_USER_DO_PPRINTF(  
    fmt,  
    ... )
```

Value:

```
if (DEBUG_USER_M_VAR) \
{ \
    printf(fmt, ##__VA_ARGS__); \
}
```

function used by the library to print debug msgs

Definition at line 21 of file ums_user_library.h.

5.4.2.2 DEBUG_USER_DO_PRINTF

```
#define DEBUG_USER_DO_PRINTF(  
    fmt )
```

Value:

```
if (DEBUG_USER_M_VAR) \
{ \
    printf(fmt); \
}
```

function used by the library to print debug msgs

Definition at line 30 of file ums_user_library.h.

5.4.2.3 ERR_USER_EXEC_PPRINTF

```
#define ERR_USER_EXEC_PPRINTF(  
    fmt,  
    ... )
```

Value:

```
if (ERR_USER_EXEC_M_VAR) \
    printf(fmt, ##__VA_ARGS__);
```

function used by the library to print err_exec msgs

Definition at line 58 of file ums_user_library.h.

5.4.2.4 ERR_USER_EXEC_PRINTF

```
#define ERR_USER_EXEC_PRINTF(  
    fmt )
```

Value:

```
    if (ERR_USER_EXEC_M_VAR)    \  
        printf(fmt);
```

function used by the library to print err_exec msgs

Definition at line 66 of file ums_user_library.h.

5.4.2.5 ERR_USER_PPRINTF

```
#define ERR_USER_PPRINTF(  
    fmt,  
    ... )
```

Value:

```
    if (ERR_USER_M_VAR)    \  
        printf(fmt, ##__VA_ARGS__);
```

function used by the library to print err msgs

Definition at line 40 of file ums_user_library.h.

5.4.2.6 ERR_USER_PRINTF

```
#define ERR_USER_PRINTF(  
    fmt )
```

Value:

```
    if (ERR_USER_M_VAR)    \  
        printf(fmt);
```

function used by the library to print err msgs

Definition at line 48 of file ums_user_library.h.

5.4.3 Function Documentation

5.4.3.1 add_worker_thread()

```
int add_worker_thread (  
    int fd,  
    int(*) (void *) work,  
    void * args,  
    ums_pid_t completion_list_id )
```

Create and add a worker thread to the desired completion list.

Parameters

<i>fd</i>	File descriptor related to the UMS
<i>work</i>	pointer to a function that will be executed by the worker thread
<i>args</i>	argument that will be passed to work()
<i>completion_list↔ _id</i>	unsigned long identifier for the completion list who will own this worker

Returns

int 0 on success, -1 otherwise

Definition at line 388 of file ums_user_library.c.

5.4.3.2 create_completion_list()

```
completion_list_data* create_completion_list (
    int fd )
```

Create a completion list object, the ID is located in create_completion_list->id, in the case of readers/writers or publish/subscribe or master/slaves use 2 different **completion_list** (p. ??) for the two groups of workers.

Parameters

<i>fd</i>	File descriptor related to the UMS
-----------	------------------------------------

Returns

completion_list_data* on success, NULL otherwise

Definition at line 371 of file ums_user_library.c.

5.4.3.3 dequeue_ums_completion_list_items()

```
completion_list_data* dequeue_ums_completion_list_items (
    int fd )
```

called from the scheduler thread obtains a list of current available thread to be run, if no thread is available to be run the function should be blocking until a thread becomes available

Parameters

<i>fd</i>	File descriptor related to the UMS
-----------	------------------------------------

Returns

struct completionm_list_data* pointer to a **completion_list_data** (p. ??)

Definition at line 192 of file ums_user_library.c.

5.4.3.4 destroy_comp_list()

```
int destroy_comp_list (
    int fd,
    ums_pid_t completion_list_id )
```

destroy the selected completion list

Parameters

<i>fd</i>	File descriptor related to the UMS
<i>completion_list↔ _id</i>	ID of the completion list to destroy

Returns

int 0 on success, -1 otherwise

Definition at line 413 of file ums_user_library.c.

5.4.3.5 enter_ums_scheduling_mode()

```
int enter_ums_scheduling_mode (
    int fd,
    int(*) (void *) entry_point,
    void * entrypoint_args,
    ums_pid_t completion_list_id )
```

Converts a standard pthread in a UMS Scheduler thread, the function takes as input a file descriptor, a completion list ID of worker threads and a entry point function.

Parameters

<i>fd</i>	File descriptor related to the UMS scheduler
<i>entry_point</i>	Pointer to a Scheduling function
<i>completion_list↔ _id</i>	ID of a completion list that will be schedule on this UMS Scheduler thread
<i>entrypoint_args</i>	pointer to entry_point funtion's parameters

Returns

int returns 0 on success, -1 otherwise

Definition at line 155 of file ums_user_library.c.

5.4.3.6 execute_ums_thread()

```
int execute_ums_thread (
    int fd,
    ums_pid_t worker_id,
    completion_list_data * list_data )
```

called from a scheduler thread, it executes the passed worker thread by switching the entire context, if -EACCES or -EBADF are returned try again with a different worker_id.

Parameters

<i>fd</i>	File descriptor related to the UMS
<i>worker↔ _id</i>	id of the worker to be executed, the id is contenuto in the struct ums_worker_thread_data (p. ??) contenua in the struct completion_list_data->head
<i>list_data</i>	pointer to the completion_list (p. ??) retrived from the dequeue, this list will be free

Returns

int returns job's return value on success, -EBADF or -EACCES if a worker is already running, err otherwise

Definition at line 256 of file ums_user_library.c.

Here is the call graph for this function:

**5.4.3.7 exit_worker_thread()**

```
int exit_worker_thread (
    int fd )
```

This function must be called at the end of every **worker_thread** (p. ??).

Parameters

<i>fd</i>	File descriptor related to the UMS
-----------	------------------------------------

Returns

int return 0 on success, -1 otherwise

Definition at line 336 of file ums_user_library.c.

Here is the caller graph for this function:

**5.4.3.8 release_ums()**

```
int release_ums (  
    int fd )
```

destroy the UMS

Parameters

<i>fd</i>	File descriptor related to the UMS
-----------	------------------------------------

Returns

int 0 on success, -1 otherwise

Definition at line 429 of file ums_user_library.c.

5.4.3.9 ums_close()

```
int ums_close (  
    int fd )
```

Release all the structures related to the fd UMS.

Returns

int return 0 on success, -1 otherwise

Definition at line 136 of file ums_user_library.c.

5.4.3.10 ums_init()

```
int ums_init ( )
```

Initialize a new UMS.

Returns

int returns the file descriptor of the device driver, -1 otherwise

Definition at line 124 of file ums_user_library.c.

5.4.3.11 ums_thread_yield()

```
int ums_thread_yield (
    int fd )
```

called from a worker thread, it pauses the execution of the current thread and the UMS scheduler entry point is executed for determining the next thread to be scheduled;

Parameters

<i>fd</i>	File descriptor related to the UMS
-----------	------------------------------------

Returns

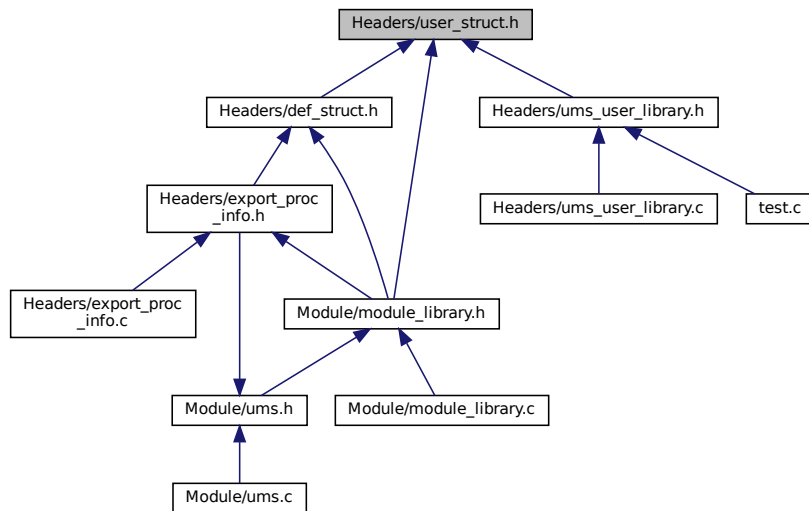
int return 0 on success, -1 otherwise

Definition at line 296 of file ums_user_library.c.

5.5 Headers/user_struct.h File Reference

Contains the definitions of every custom type used in user space.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct **sched_thread_data**
*Struct used to share **sched_thread_data** (p. ??) info with the kernel.*
- struct **completion_list_data**
*Struct used to share **completion_list_data** (p. ??) info with the kernel.*
- struct **ums_worker_thread_data**
*Struct used to share **ums_worker_thread_data** (p. ??) infos with the kernel.*

Typedefs

- typedef unsigned long long **ums_pid_t**
- typedef struct **sched_thread_data** **sched_thread_data**
*Struct used to share **sched_thread_data** (p. ??) info with the kernel.*
- typedef struct **completion_list_data** **completion_list_data**
*Struct used to share **completion_list_data** (p. ??) info with the kernel.*
- typedef struct **ums_worker_thread_data** **ums_worker_thread_data**
*Struct used to share **ums_worker_thread_data** (p. ??) infos with the kernel.*

5.5.1 Detailed Description

Contains the definitions of every custom type used in user space.

5.5.2 Typedef Documentation

5.5.2.1 completion_list_data

```
typedef struct completion_list_data completion_list_data
```

Struct used to share **completion_list_data** (p. ??) info with the kernel.

5.5.2.2 sched_thread_data

```
typedef struct sched_thread_data sched_thread_data
```

Struct used to share **sched_thread_data** (p. ??) info with the kernel.

5.5.2.3 ums_worker_thread_data

```
typedef struct ums_worker_thread_data ums_worker_thread_data
```

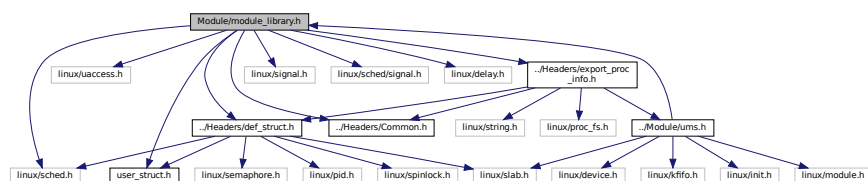
Struct used to share **ums_worker_thread_data** (p. ??) infos with the kernel.

5.6 Module/module_library.h File Reference

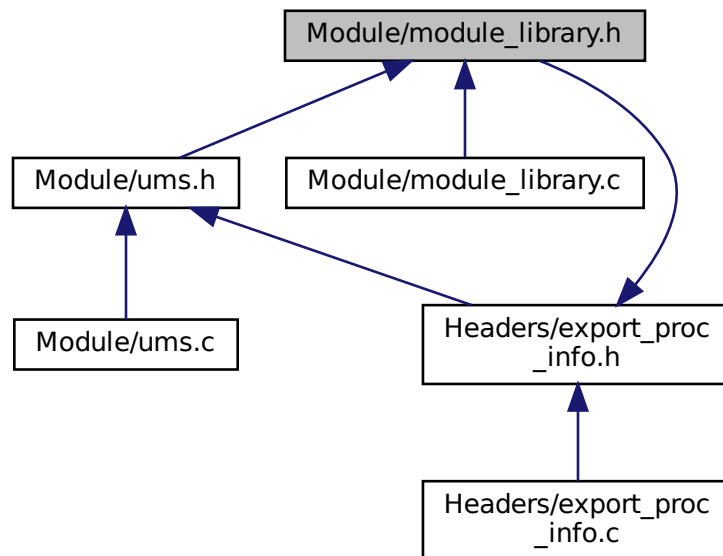
Contains the helper functions called from the UMS device.

```
#include <linux/sched.h>
#include <linux/uaccess.h>
#include "../Headers/def_struct.h"
#include "../Headers/Common.h"
#include <linux/signal.h>
#include "../Headers/user_struct.h"
#include <linux/sched/signal.h>
#include <linux/delay.h>
#include "../Headers/export_proc_info.h"
```

Include dependency graph for module_library.h:



This graph shows which files directly or indirectly include this file:



Functions

- int **create_ums_thread** (struct list_head *sched_data, unsigned long int filep, int(*entry_point)(void *), void * args, struct list_head *comp_lists, ums_pid_t _completion_list)
Create a new **scheduler_thread** (p. ??) struct, by relating entry_point and _completion_list.
- struct **completion_list** * **retrive_current_list** (struct list_head *sched_data, unsigned long int filep)
retrive the completion list managed by the current **scheduler_thread** (p. ??)
- struct **scheduler_thread** * **retrive_current_scheduler_thread** (struct list_head *sched_data, unsigned long int filep)
Retrive the current **scheduler_thread** (p. ??) structure.
- int **release_scheduler** (struct list_head *sched_data, unsigned long int filep)
release the current UMS Scheduler and the relative compeltion_lists(if possible) and scheduler_threads, if the schedulers are currently executing a worker this function returs -5;
- struct **completion_list** * **retrive_list** (struct list_head *comp_lists, ums_pid_t id)
Retrive the desired **completion_list** (p. ??).
- struct **worker_thread** * **retrive_worker_thread** (struct **completion_list** *current_list, ums_pid_t pid)
retrive the desired **worker_thread** (p. ??)
- int **list_copy_to_user** (struct **completion_list_data** *to_write, struct **completion_list** *from_read)
Create a copy of the current completion list for the user.
- int **del_worker_thread** (struct list_head worker_list, struct **worker_thread** *worker_to_del)
free the desired worker
- char * **_retrive_worker_thread_proc** (time64_t pid, struct list_head *comp_lists)
retrive the infos form the worker's struct in a char*
- char * **_retrive_schduler_thread_proc** (time64_t pid, struct list_head *sched_data)
retrive the infos form the schduler_thread's struct in a char*
- int **destoy_comp_list_set_destroy** (unsigned long arg, struct list_head *comp_lists)

- Destroy the choosen comp list with id:completion_list_id.*
- int **add_worker** (unsigned long arg, ums_pid_t worker_counter, struct list_head *comp_lists)
*This function generate a new **worker_thread** (p. ??) form **ums_worker_thread_data** (p. ??) and then add it to the desired **completion_list** (p. ??).*
 - int **exit_worker_thread** (struct list_head *sched_data, unsigned long int filep, unsigned long arg)
This functions delete the worker and pass to the user new entry_point function.
 - int **exit_from_yield** (struct list_head *sched_data, unsigned long int filep)
This function update the running time of thw worker and decrement the nesting variable.
 - int **ums_thread_yield** (struct list_head *sched_data, unsigned long int filep, unsigned long arg)
Pause the execution of the current worker and write in (sched_thread_data)arg entry_point and args.*
 - int **_execute_ums_thread** (struct list_head *sched_data, unsigned long int filep, unsigned long arg, struct **scheduler_thread** *cur_scheduler, struct **scheduler_thread_worker_lifo** *worker_lifo_aux)
Retrive from the user te id of the worker to execute, if found save the worker in the UMS Scheduler thread list of worker and pass to the user the work and args.
 - int **execute_ums_thread** (struct list_head *sched_data, unsigned long int filep, unsigned long arg, struct **scheduler_thread_worker_lifo** *worker_lifo_aux)
this functions is a wrapper for _execute_ums_thread, to manage the data statistics
 - int **dequeue_size_request** (struct list_head *sched_data, unsigned long int filep, unsigned long arg)
*This function return the size to be allocated in user space in order to recive the list of **ums_worker_thread_data(user space struct)** (p. ??)*
 - int **enter_ums_scheduling_mode** (struct list_head *sched_data, unsigned long int filep, unsigned long arg, struct list_head *comp_lists)
This function converts a standard pthread in a UMS Scheduler thread.

5.6.1 Detailed Description

Contains the helper functions called from the UMS device.

5.6.2 Function Documentation

5.6.2.1 _execute_ums_thread()

```
int _execute_ums_thread (
    struct list_head * sched_data,
    unsigned long int filep,
    unsigned long arg,
    struct scheduler_thread * cur_scheduler,
    struct scheduler_thread_worker_lifo * worker_lifo_aux )
```

Retrive from the user te id of the worker to execute, if found save the worker in the UMS Scheduler thread list of worker and pass to the user the work and args.

Parameters

<i>sched_data</i>	pointer to a UMS Scheduler list_head who manage this scheduler thread
<i>filep</i>	id of the UMS Scheduler
<i>arg</i>	pointer to a user space sched_thread_data* passed with the id of the choosed woker, than will be filled with entry_point and *args
<i>worker_lifo_aux</i>	structure that will contain the choosed wroker to be added to the UMS Scheduler thread
<i>cur_scheduler</i>	pointer to the current UMS Scheduler thread

Returns

int 0 on success, err otherwise. err:

- -EFBIG, no more space in kernel to allow a new **scheduler_thread_worker_lifo** (p. ??)
- -EBADF, worker not found
- -EACCES, the worker is not RUNNABLE with one of this errno the default operation is to try to execute the next worker;

Definition at line 939 of file module_library.c.

5.6.2.2 _retrive_scheduler_thread_proc()

```
char* _retrive_scheduler_thread_proc (
    time64_t pid,
    struct list_head * sched_data )
```

retrive the infos form the schduler_thread's struct in a char*

Parameters

<i>pid</i>	the selected schduler_thread id
<i>sched_data</i>	pointer to a UMS Scheduler list_head where the scheduler thread is stored

Returns

char* !=NULL on success, NULL otherwise

Definition at line 567 of file module_library.c.

Here is the caller graph for this function:

**5.6.2.3 _retrive_worker_thread_proc()**

```
char* _retrive_worker_thread_proc (
    time64_t pid,
    struct list_head * comp_lists )
```

retrive the infos form the worker's struct in a char*

Parameters

<i>pid</i>	the selected worker id
<i>comp_lists</i>	pointer to a Completion list list_head where the worker is stored

Returns

char* !=NULL on success, NULL otherwise

Definition at line 606 of file module_library.c.

Here is the caller graph for this function:



5.6.2.4 add_worker()

```

int add_worker (
    unsigned long arg,
    ums_pid_t worker_counter,
    struct list_head * comp_lists )

```

This function generate a new **worker_thread** (p. ??) form **ums_worker_thread_data** (p. ??) and then add it to the desired **completion_list** (p. ??).

Parameters

<i>arg</i>	pointer to a user space ums_worker_thread_data (p. ??) * that contains the new woker infos
<i>worker_to_add</i>	worker passed from user space
<i>new_worker_thread</i>	struct allocated from the kernel to be filled
<i>worker_counter</i>	monotonic counter of the worker
<i>comp_lists</i>	pointer to the completion_list (p. ??) list_head who manage the choosen completion_list (p. ??)

Returns

int 0 on succes, err otherwise,if err=EACCES the completion list is marked to be destroyed.

Definition at line 697 of file module_library.c.

5.6.2.5 create_ums_thread()

```
int create_ums_thread (
    struct list_head * sched_data,
    unsigned long int filep,
    int(*) (void *) entry_point,
    void * args,
    struct list_head * comp_lists,
    ums_pid_t _completion_list )
```

Create a new **scheduler_thread** (p. ??) struct, by relating entry_point and _completion_list.

Parameters

<i>sched_data</i>	pointer to the UMS Scheduler global variable that manage all the UMS Scheduler
<i>filep</i>	identifier of the file descriptoy
<i>entry_point</i>	scheduling function
<i>args</i>	pointer to the args to pass to the entry_point function
<i>comp_lists</i>	pointer to the UMS Scheduler global variable that manage all the completion lists
<i>_completion_list</i>	identifier of the completion list to relate to this scheduler thread

Returns

int 0 on success err otherwise for the possible err values, they are same of the relative ioctl

Definition at line 106 of file module_library.c.

5.6.2.6 del_worker_thread()

```
int del_worker_thread (
    struct list_head worker_list,
    struct worker_thread * worker_to_del )
```

free the desired worker

Parameters

<i>worker_list</i>	the list_head who own this worker
<i>worker_to_del</i>	pointer to the worker to del

Returns

int 0 on success, -1 otherwise

Definition at line 536 of file module_library.c.

5.6.2.7 dequeue_size_request()

```
int dequeue_size_request (
    struct list_head * sched_data,
    unsigned long int filep,
    unsigned long arg )
```

This function return the size to be allocated in user space in order to recive the list of **ums_worker_thread_↔data(user space struct)** (p. ??)

Parameters

<i>sched_data</i>	pointer to a UMS Scheduler list_head who manage this scheduler thread
<i>filep</i>	id of the UMS Scheduler
<i>arg</i>	pointer to a user space completion_list_data* used to set the requeste_size var

Returns

int 0 on success, err otherwise

Definition at line 1081 of file module_library.c.

Here is the call graph for this function:



5.6.2.8 destroy_comp_list_set_destroy()

```
int destroy_comp_list_set_destroy (
    unsigned long arg,
    struct list_head * comp_lists )
```

Destroy the choosen comp list with id:completion_list_id.

Parameters

<i>arg</i>	pointer to teh completion_list_data* that contains the info of the choosed completion list
<i>comp_lists</i>	pointer to the list_head who own all the completion_list (p. ??) for this UMS Scheduler
<i>completion_list_↔_id</i>	the id of the completion list to destroy

Returns

int 0 on success, err otherwise, if err=ENOMEM no more space in the kernel

Definition at line 668 of file module_library.c.

5.6.2.9 enter_ums_scheduling_mode()

```
int enter_ums_scheduling_mode (
    struct list_head * sched_data,
    unsigned long int filep,
    unsigned long arg,
    struct list_head * comp_lists )
```

This function converts a standard pthread in a UMS Scheduler thread.

Parameters

<i>sched_data</i>	pointer to a UMS Scheduler list_head who manage this scheduler thread
<i>filep</i>	id of the UMS Scheduler
<i>arg</i>	pointer to a user space sched_thread_data* used to read the information needed to convert a pthread into a UMS Scheduler threads
<i>comp_lists</i>	pointer to the completion_list (p. ??) list_head who manage the choosen completion_list (p. ??)

Returns

0 on success, err otherwise

Definition at line 1122 of file module_library.c.

5.6.2.10 execute_ums_thread()

```
int execute_ums_thread (
    struct list_head * sched_data,
    unsigned long int filep,
    unsigned long arg,
    struct scheduler_thread_worker_lifo * worker_lifo_aux )
```

this functions is a wrapper for _execute_ums_thread, to manage the data statistics

Parameters

<i>sched_data</i>	pointer to a UMS Scheduler list_head who manage this scheduler thread
<i>filep</i>	id of the UMS Scheduler
<i>arg</i>	pointer to a user space sched_thread_data* passed with the id of the choosed woker, than will be filled with entry_point and *args
<i>worker_lifo_aux</i>	structure that will contain the choosed wroker to be added to the UMS Scheduler thread

Returns

int 0 on success, err otherwise. err:

- -EFBIG, no more space in kernel to allow a new **scheduler_thread_worker_lifo** (p. ??)
- -EBADF, worker not found
- -EACCES, the worker is not RUNNABLE with one of this errno the default operation is to try to execute the next worker;

Definition at line 1024 of file module_library.c.

5.6.2.11 exit_from_yield()

```
int exit_from_yield (
    struct list_head * sched_data,
    unsigned long int filep )
```

This function update the running time of thw worker and decrement the nesting variable.

Parameters

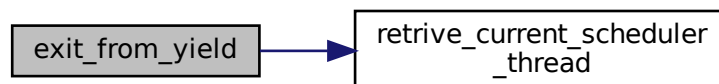
<i>sched_data</i>	pointer to a UMS Scheduler list_head who manage this scheduler thread
<i>filep</i>	id of the UMS Scheduler

Returns

int 0 on success, err otherwise

Definition at line 863 of file module_library.c.

Here is the call graph for this function:

**5.6.2.12 exit_worker_thread()**

```
int exit_worker_thread (
    struct list_head * sched_data,
    unsigned long int filep,
    unsigned long arg )
```

This functions delete the worker and pass to the user new entry_point function.

Parameters

<i>sched_data</i>	pointer to a UMS Scheduler list_head who manage this scheduler thread
<i>filep</i>	id of the UMS Scheduler
<i>arg</i>	pointer passed fro user space

Returns

int 0 on success, err otherwise, if err=-EBUSY :no other worker to execute

Definition at line 761 of file module_library.c.

5.6.2.13 list_copy_to_user()

```
int list_copy_to_user (
    struct completion_list_data * to_write,
    struct completion_list * from_read )
```

Create a copy of the current completion list for the user.

Parameters

<i>to_write</i>	pointer to a completion_list_data (p. ??) struct, given by the user
<i>from_read</i>	pointer to a completion_list (p. ??) struct, given by the kernel

Returns

int 0 on success, err otherwise

Definition at line 443 of file module_library.c.

5.6.2.14 release_scheduler()

```
int release_scheduler (
    struct list_head * sched_data,
    unsigned long int filep )
```

release the current UMS Scheduler and the relative compeltion_lists(if possible) and scheduler_threads, if the schedulers are currently executing a worker this function returs -5;

Parameters

<i>sched_data</i>	pointer to the UMS Scheduler global variable that manage all the UMS Scheduler
<i>filep</i>	identifier of the file descriptoy

Returns

int 0 on success err otherwise

Definition at line 291 of file module_library.c.

5.6.2.15 retriave_current_list()

```
struct completion_list* retriave_current_list (  
    struct list_head * sched_data,  
    unsigned long int filep )
```

retriave the completion list managed by the current **scheduler_thread** (p. ??)

Parameters

<i>sched_data</i>	pointer to the UMS Scheduler global variable that manage all the UMS Scheduler
<i>filep</i>	identifier of the file descriptoy

Returns

struct completion_list* on success NULL otherwise

Definition at line 167 of file module_library.c.

Here is the caller graph for this function:

**5.6.2.16 retriave_current_scheduler_thread()**

```
struct scheduler_thread* retriave_current_scheduler_thread (  
    struct list_head * sched_data,  
    unsigned long int filep )
```

Retrive the current **scheduler_thread** (p. ??) structure.

Parameters

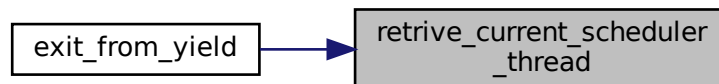
<i>sched_data</i>	pointer to the UMS Scheduler global variable that manage all the UMS Scheduler
<i>filep</i>	identifier of the file descriptoy

Returns

struct scheduler_thread* on success NULL otherwise

Definition at line 235 of file module_library.c.

Here is the caller graph for this function:

**5.6.2.17 retrieve_list()**

```

struct completion_list* retrieve_list (
    struct list_head * comp_lists,
    ums_pid_t id )
  
```

Retrive the desired **completion_list** (p. ??).

Parameters

<i>comp_lists</i>	pointer to the UMS Scheduler global variable that manage all the completion lists
<i>id</i>	id of the completion_list (p. ??) choosen

Returns

struct completion_list* on success, NULL otherwise

Definition at line 398 of file module_library.c.

5.6.2.18 retrieve_worker_thread()

```
struct worker_thread* retrieve_worker_thread (
    struct completion_list * current_list,
    ums_pid_t pid )
```

retrieve the desired **worker_thread** (p. ??)

Parameters

<i>current_list</i>	pointer to the scheduler_thread-> completion_list (p. ??)
<i>pid</i>	id of the worker_thread (p. ??)

Returns

struct **worker_thread*** on success, NULL otherwise

Definition at line 424 of file module_library.c.

5.6.2.19 ums_thread_yield()

```
int ums_thread_yield (
    struct list_head * sched_data,
    unsigned long int filep,
    unsigned long arg )
```

Pause the execution of the current worker and write in (sched_thread_data*)arg entry_point and args.

Parameters

<i>sched_data</i>	pointer to a UMS Scheduler list_head who manage this scheduler thread
<i>filep</i>	id of the UMS Scheduler
<i>arg</i>	pointer to a user space sched_thread_data* taht will be fill with entry_point and *args

Returns

errno

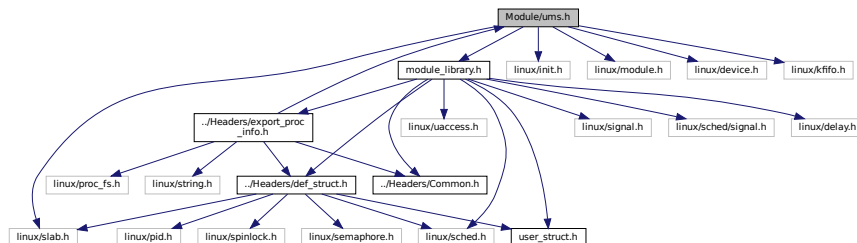
- errno:
 - EBUSY, no other workers, this mean that you should just call the EXIT_FROM_YIELD
 - EAGAIN, worker found, you have to call the entryptoint before call the EXIT_FROM_YIELD

Definition at line 891 of file module_library.c.

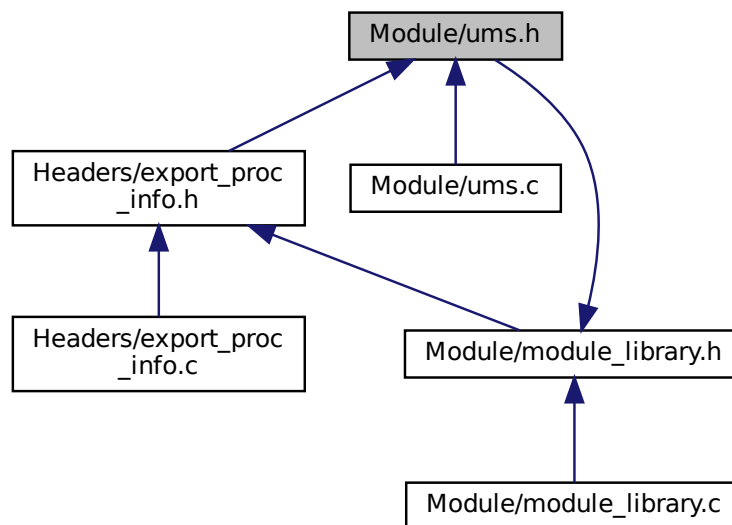
5.7 Module/ums.h File Reference

Contains the functions to interact from user space with the kernel module.

```
#include <linux/slab.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/device.h>
#include <linux/kfifo.h>
#include "module_library.h"
Include dependency graph for ums.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- int **ums_dev_open** (struct inode *inodep, struct file *filep)
Used to create a new UMS scheduler.
- int **ums_dev_release** (struct inode *inodep, struct file *filep)
Function not implemented, to close call the ioctl(DESTROY_COMP_LIST) and then ioctl(RELEASE_UMS)
- long **ums_dev_ioctl** (struct file *filep, unsigned int cmd, unsigned long arg)
*This function wraps: all the ioctls declared in **Common.h** (p. ??).*
- char * **retrive_worker_thread_proc** (time64_t pid)

- retrives the info of a worker thread and puts it in a buffer*
- `char * retrive_scheduler_thread_proc (time64_t pid)`
retrive the info of a schduler thread and puts it in a buffer

5.7.1 Detailed Description

Contains the functions to interact from user space with the kernel module.

5.7.2 Function Documentation

5.7.2.1 `retrive_scheduler_thread_proc()`

```
char* retrive_scheduler_thread_proc (
    time64_t pid )
```

retrive the info of a schduler thread and puts it in a buffer

Parameters

<code>pid</code>	the selected schduler thread's id
------------------	-----------------------------------

Returns

`char* !=NULL` on success, `NULL` otherwise

Definition at line 743 of file ums.c.

Here is the call graph for this function:



5.7.2.2 `retrive_worker_thread_proc()`

```
char* retrive_worker_thread_proc (
    time64_t pid )
```

retrives the info of a worker thread and puts it in a buffer

Parameters

<i>pid</i>	the selected worker id
------------	------------------------

Returns

char* !=NULL on success, NULL otherwise

Definition at line 655 of file ums.c.

Here is the call graph for this function:

**5.7.2.3 ums_dev_ioctl()**

```

long ums_dev_ioctl (
    struct file * filep,
    unsigned int cmd,
    unsigned long arg )
  
```

This function wraps: all the ioctls declared in **Common.h** (p. ??).

Parameters

<i>filep</i>	
<i>cmd</i>	the IOCTL defined in Common.h (p. ??)
<i>arg</i>	used to copy_from/copy_to user space

Returns

long

ENTER_UMS_SCHEDULING_MODE

reads from args *sched_thread_data with: -entrypoint **-args(for the entry point)** (p. ??) -the completion list id

Then call create_ums_thread.

returns 0 on success, err otherwise, the errno is setted

UMS_THREAD_YIELD

called from a worker thread, it pauses the execution of the current thread and the UMS scheduler entry point is executed for determining the next thread to be scheduled;

Takes as input `*sched_thread_data` fill it with `*entry_point` and `*args` The worker will be set to YIELD

errno:

- EBUSY, no other workers, this mean that you should just call the EXIT_FROM_YIELD
- EAGAIN, worker found, you have to call the entryptpoint before call the EXIT_FROM_YIELD

Definition at line 195 of file ums.c.

5.7.2.4 ums_dev_open()

```
int ums_dev_open (
    struct inode * inodep,
    struct file * filep )
```

Used to create a new UMS scheduler.

Parameters

<i>inodep</i>	
<i>filep</i>	

Returns

int 0 on success, err otherwise

Definition at line 125 of file ums.c.

5.7.2.5 ums_dev_release()

```
int ums_dev_release (
    struct inode * inodep,
    struct file * filep )
```

Function not implemented, to close call the `ioctl(DESTROY_COMP_LIST)` and then `ioctl(RELEASE_UMS)`

Parameters

<i>inodep</i>	
<i>filep</i>	

Returns

int 0 on success, err otherwise

Definition at line 186 of file ums.c.

Index

- `_execute_ums_thread`
 - `module_library.h`, 53
 - `_retrive_scheduler_thread_proc`
 - `module_library.h`, 54
 - `_retrive_worker_thread_proc`
 - `module_library.h`, 54
- `ADD_WORKER`
 - `Common.h`, 26
- `add_worker`
 - `module_library.h`, 55
- `add_worker_thread`
 - `ums_user_library.h`, 44
- `args`, 7
 - `sched_thread_data`, 13
 - `scheduler_thread`, 15
 - `ums_worker_thread_data`, 21
 - `worker_thread`, 23
- `Common.h`
 - `ADD_WORKER`, 26
 - `CREATE_COMP_LIST`, 27
 - `DEBUG_DO_PPRINTK`, 27
 - `DEBUG_DO_PRINTK`, 28
 - `DEQUEUE_SIZE_REQUEST`, 28
 - `DEQUEUE_UMS_COMPLETION_LIST_ITEMS`, 28
 - `DESTROY_COMP_LIST`, 29
 - `ENTER_UMS_SCHEDULING_MODE`, 29
 - `ERR_PPRINTK`, 30
 - `ERR_PRINTK`, 30
 - `EXECUTE_UMS_THREAD`, 30
 - `EXIT_FROM_YIELD`, 31
 - `EXIT_WORKER_THREAD`, 31
 - `NOTIFY_EXEC_ERR_PPRINTK`, 32
 - `RELEASE_UMS`, 32
 - `UMS_THREAD_YIELD`, 32
- `completion_list`, 7
 - `def_struct.h`, 35
 - `id`, 8
 - `list_lock`, 8
 - `sched_thread_data`, 13
 - `sem_counter`, 8
 - `to_destroy`, 8
 - `workers`, 8
- `completion_list_data`, 9
 - `head`, 10
 - `id`, 10
 - `requested_size`, 10
 - `user_struct.h`, 50
- `completion_list_id`
 - `ums_worker_thread_data`, 21
 - `worker_thread`, 23
- `CREATE_COMP_LIST`
 - `Common.h`, 27
- `create_completion_list`
 - `ums_user_library.h`, 45
- `create_proc_ums`
 - `export_proc_info.h`, 37
- `create_proc_ums_scheduler`
 - `export_proc_info.h`, 38
- `create_proc_ums_scheduler_thread`
 - `export_proc_info.h`, 38
- `create_proc_worker`
 - `export_proc_info.h`, 39
- `create_ums_thread`
 - `module_library.h`, 55
- `cur_size`
 - `dequeue_fast_list`, 11
- `currently_running`
 - `scheduler_thread`, 15
- `DEBUG_DO_PPRINTK`
 - `Common.h`, 27
- `DEBUG_DO_PRINTK`
 - `Common.h`, 28
- `DEBUG_USER_DO_PPRINTF`
 - `ums_user_library.h`, 42
- `DEBUG_USER_DO_PRINTF`
 - `ums_user_library.h`, 43
- `def_struct.h`
 - `completion_list`, 35
 - `scheduler_thread`, 35
 - `scheduler_thread_worker_lifo`, 35
 - `UMS_sched_data`, 35
 - `worker_thread`, 36
- `del_worker_thread`
 - `module_library.h`, 56
- `dequeue_fast_list`, 11
 - `cur_size`, 11
 - `list_data`, 11
 - `list_pointer`, 12
 - `origin_size`, 12
- `DEQUEUE_SIZE_REQUEST`
 - `Common.h`, 28
- `dequeue_size_request`
 - `module_library.h`, 56
- `DEQUEUE_UMS_COMPLETION_LIST_ITEMS`
 - `Common.h`, 28
- `dequeue_ums_completion_list_items`

- ums_user_library.h, 45
- destroy_comp_list_set_destroy
 - module_library.h, 57
- destroy
 - scheduler_thread, 15
- DESTROY_COMP_LIST
 - Common.h, 29
- destroy_comp_list
 - ums_user_library.h, 46
- ended
 - scheduler_thread, 15
- ent
 - scheduler_thread, 15
 - UMS_sched_data, 19
- ent_write
 - scheduler_thread, 15
 - UMS_sched_data, 19
- ENTER_UMS_SCHEDULING_MODE
 - Common.h, 29
- enter_ums_scheduling_mode
 - module_library.h, 58
 - ums_user_library.h, 46
- entry_point
 - sched_thread_data, 13
 - scheduler_thread, 16
- ERR_PPRINTK
 - Common.h, 30
- ERR_PRINTK
 - Common.h, 30
- ERR_USER_EXEC_PPRINTF
 - ums_user_library.h, 43
- ERR_USER_EXEC_PRINTF
 - ums_user_library.h, 43
- ERR_USER_PPRINTF
 - ums_user_library.h, 44
- ERR_USER_PRINTF
 - ums_user_library.h, 44
- execute_lock
 - worker_thread, 23
- EXECUTE_UMS_THREAD
 - Common.h, 30
- execute_ums_thread
 - module_library.h, 58
 - ums_user_library.h, 47
- EXIT_FROM_YIELD
 - Common.h, 31
- exit_from_yield
 - module_library.h, 59
- EXIT_WORKER_THREAD
 - Common.h, 31
- exit_worker_thread
 - module_library.h, 59
 - ums_user_library.h, 47
- export_proc_info.h
 - create_proc_ums, 37
 - create_proc_ums_scheduler, 38
 - create_proc_ums_scheduler_thread, 38
 - create_proc_worker, 39
 - rm_proc_ums, 39
 - rm_proc_ums_scheduler, 39
 - rm_proc_ums_scheduler_thread, 40
 - rm_proc_worker, 40
- father
 - scheduler_thread, 16
 - UMS_sched_data, 20
- head
 - completion_list_data, 10
- Headers/Common.h, 25
- Headers/def_struct.h, 33
- Headers/export_proc_info.h, 36
- Headers/ums_user_library.h, 41
- Headers/user_struct.h, 49
- id
 - completion_list, 8
 - completion_list_data, 10
- id_counter
 - UMS_sched_data, 20
- last_switch_time
 - scheduler_thread, 16
- list_copy_to_user
 - module_library.h, 60
- list_data
 - dequeue_fast_list, 11
- list_lock
 - completion_list, 8
- list_pointer
 - dequeue_fast_list, 12
- mean_switch_time
 - scheduler_thread, 16
- Module/module_library.h, 51
- Module/ums.h, 63
- module_library.h
 - _execute_ums_thread, 53
 - _retrive_scheduler_thread_proc, 54
 - _retrive_worker_thread_proc, 54
 - add_worker, 55
 - create_ums_thread, 55
 - del_worker_thread, 56
 - dequeue_size_request, 56
 - destroy_comp_list_set_destroy, 57
 - enter_ums_scheduling_mode, 58
 - execute_ums_thread, 58
 - exit_from_yield, 59
 - exit_worker_thread, 59
 - list_copy_to_user, 60
 - release_scheduler, 60
 - retrive_current_list, 61
 - retrive_current_scheduler_thread, 61
 - retrive_list, 62
 - retrive_worker_thread, 62
 - ums_thread_yield, 63
- nesting

- scheduler_thread, 16
- next
 - ums_worker_thread_data, 22
- NOTIFY_EXEC_ERR_PPRINTK
 - Common.h, 32
- numb_switch
 - scheduler_thread, 17
 - worker_thread, 23
- origin_size
 - dequeue_fast_list, 12
- owner
 - UMS_sched_data, 20
- pid
 - scheduler_thread, 17
 - ums_worker_thread_data, 22
 - worker_thread, 24
- release_scheduler
 - module_library.h, 60
- RELEASE_UMS
 - Common.h, 32
- release_ums
 - ums_user_library.h, 48
- requested_size
 - completion_list_data, 10
- retrive_current_list
 - module_library.h, 61
- retrive_current_scheduler_thread
 - module_library.h, 61
- retrive_list
 - module_library.h, 62
- retrive_scheduler_thread_proc
 - ums.h, 65
- retrive_worker_thread
 - module_library.h, 62
- retrive_worker_thread_proc
 - ums.h, 65
- rm_proc_ums
 - export_proc_info.h, 39
- rm_proc_ums_scheduler
 - export_proc_info.h, 39
- rm_proc_ums_scheduler_thread
 - export_proc_info.h, 40
- rm_proc_worker
 - export_proc_info.h, 40
- sched_thread_data, 12
 - args, 13
 - completion_list, 13
 - entry_point, 13
 - user_struct.h, 51
- scheduler_thread, 14
 - args, 15
 - currently_running, 15
 - def_struct.h, 35
 - destroy, 15
 - ended, 15
 - ent, 15
 - ent_write, 15
 - entry_point, 16
 - father, 16
 - last_switch_time, 16
 - mean_switch_time, 16
 - nesting, 16
 - numb_switch, 17
 - pid, 17
 - started, 17
 - UMS_sched_data, 20
 - yield_priority, 17
- scheduler_thread_worker_lifo, 18
 - def_struct.h, 35
- sem_counter
 - completion_list, 8
- started
 - scheduler_thread, 17
- state
 - worker_thread, 24
- task_struct
 - worker_thread, 24
- to_destroy
 - completion_list, 8
- ums.h
 - retrive_scheduler_thread_proc, 65
 - retrive_worker_thread_proc, 65
 - ums_dev_ioctl, 66
 - ums_dev_open, 67
 - ums_dev_release, 67
- ums_close
 - ums_user_library.h, 48
- ums_dev_ioctl
 - ums.h, 66
- ums_dev_open
 - ums.h, 67
- ums_dev_release
 - ums.h, 67
- ums_device_data_s, 18
- ums_init
 - ums_user_library.h, 49
- UMS_sched_data, 19
 - def_struct.h, 35
- ent, 19
 - ent_write, 19
 - father, 20
 - id_counter, 20
 - owner, 20
 - scheduler_thread, 20
- UMS_THREAD_YIELD
 - Common.h, 32
- ums_thread_yield
 - module_library.h, 63
 - ums_user_library.h, 49
- ums_user_library.h
 - add_worker_thread, 44
 - create_completion_list, 45

- DEBUG_USER_DO_PPRINTF, 42
- DEBUG_USER_DO_PRINTF, 43
- dequeue_ums_completion_list_items, 45
- destroy_comp_list, 46
- enter_ums_scheduling_mode, 46
- ERR_USER_EXEC_PPRINTF, 43
- ERR_USER_EXEC_PRINTF, 43
- ERR_USER_PPRINTF, 44
- ERR_USER_PRINTF, 44
- execute_ums_thread, 47
- exit_worker_thread, 47
- release_ums, 48
- ums_close, 48
- ums_init, 49
- ums_thread_yield, 49
- ums_worker_thread_data, 21
 - args, 21
 - completion_list_id, 21
 - next, 22
 - pid, 22
 - user_struct.h, 51
 - work, 22
- user_struct.h
 - completion_list_data, 50
 - sched_thread_data, 51
 - ums_worker_thread_data, 51
- work
 - ums_worker_thread_data, 22
 - worker_thread, 24
- worker_thread, 22
 - args, 23
 - completion_list_id, 23
 - def_struct.h, 36
 - execute_lock, 23
 - numb_switch, 23
 - pid, 24
 - state, 24
 - task_struct, 24
 - work, 24
- workers
 - completion_list, 8
- yield_priority
 - scheduler_thread, 17