User Mode Scheduling

0.0.1

Generated by Doxygen 1.8.17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 entry_point_args_t Struct Reference

arguments of a entry_point function

```
#include <ums_types.h>
```

Data Fields

- reason_t **reason**
- ums_context_descriptor_t activation_payload
- void * sched_args

3.1.1 Detailed Description

arguments of a entry_point function

Definition at line 64 of file ums_types.h.

3.1.2 Field Documentation

3.1.2.1 activation_payload

```
ums_context_descriptor_t activation_payload
```

reason of the scheduler call: REASON_STARTUP REASON_THREAD_YIELD REASON_THREAD_ENDED

Definition at line 69 of file ums_types.h.

3.1.2.2 sched_args

```
void* sched_args
```

if reason is yielded or ended thread, indicates the descriptor of the ums_context

Definition at line 71 of file ums_types.h.

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

- src/UMS/common/ums_types.h

3.2 idr_for_each_handler_arg_t Struct Reference

Data Fields

- char ∗ **buff**
- ssize_t **buff_size**
- int **offset**

3.2.1 Detailed Description

Definition at line 531 of file ums_scheduler.h.

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

- src/UMS/UMS_LKM/ums_scheduler.h

3.3 info_ums_context_t Struct Reference

used to choose a ums_context from the ready list or from the completion_list

```
#include <ums_types.h>
```

Data Fields

- ums_context_descriptor_t **ucd**
- unsigned int **run_time_ms**
- int **number_switch**
- void ∗ **user_reserved**
- bool **from_cl**

3.3.1 Detailed Description

used to choose a ums_context from the ready list or from the completion_list

Definition at line 78 of file ums_types.h.

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

- src/UMS/common/ums_types.h

3.4 rq_completion_list_add_remove_ums_context_args_t Struct Reference

Data Fields

- pid_t **tgid**
- ums_completion_list_descriptor_t **completion_list_d**
- ums_context_descriptor_t **ums_context_d**

3.4.1 Detailed Description

Definition at line 73 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.5 rq_create_delete_completion_list_args_t Struct Reference

Data Fields

- pid_t **tgid**
- ums_completion_list_descriptor_t **descriptor**

3.5.1 Detailed Description

Definition at line 53 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.6 rq_create_delete_process_args_t Struct Reference

Data Fields

- pid_t **tgid**

3.6.1 Detailed Description

Definition at line 46 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.7 rq_create_delete_ums_context_args_t Struct Reference

Data Fields

- pid_t **tgid**
- ums_context_descriptor_t **descriptor**
- void ∗(∗ **routine**)(void ∗args)
- void ∗ **args**
- void ∗ **user_res**
- int **cpu_core**

3.7.1 Detailed Description

Definition at line 61 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.8 rq_create_delete_ums_scheduler_args_t Struct Reference

Data Fields

- pid_t **tgid**
- pid_t **pid**
- ums_completion_list_descriptor_t **completion_list_d**
- void(∗ **entry_point_func**)(entry_point_args_t ∗ep_args)
- void ∗ **sched_args**
- entry_point_args_t ∗ **entry_point_args**
- int **return_value**
- int **cpu_core**

3.8.1 Detailed Description

Definition at line 85 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.9 rq_end_thread_args_t Struct Reference

Data Fields

- ums_context_descriptor_t **ucd**
- pid_t **pid_scheduler**

3.9.1 Detailed Description

Definition at line 131 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.10 rq_execute_args_t Struct Reference

Data Fields

- info_ums_context_t ∗ **info_context**
- pid_t **tgid**
- pid_t **pid**
- void ∗(∗ **routine**)(void ∗args)
- void ∗ **args**
- pid_t **pid_scheduler**
- ums_context_descriptor_t **ucd**
- int **cpu_core**

3.10.1 Detailed Description

Definition at line 164 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.11 rq_execute_next_new_thread_args_t Struct Reference

Data Fields

- pid_t **tgid**
- pid_t **pid**
- void ∗(∗ **routine**)(void ∗args)
- void ∗ **args**
- pid_t **pid_scheduler**
- int **cpu_core**
- ums_context_descriptor_t **ucd**

3.11.1 Detailed Description

Definition at line 101 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.12 rq_execute_next_ready_thread_args_t Struct Reference

Data Fields

- pid_t **tgid**
- pid_t **pid**

3.12.1 Detailed Description

Definition at line 116 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.13 rq_get_from_cl_args_t Struct Reference

Data Fields

- info_ums_context_t ∗ **info_context_array**
- size_t **array_size**

3.13.1 Detailed Description

Definition at line 151 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.14 rq_get_from_rl_args_t Struct Reference

Data Fields

- info_ums_context_t ∗ **info_context_array**
- size_t **array_size**

3.14.1 Detailed Description

Definition at line 158 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.15 rq_startup_new_thread_args_t Struct Reference

Data Fields

- ums_context_descriptor_t **ucd**
- pid_t **pid_scheduler**

3.15.1 Detailed Description

Definition at line 123 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.16 rq_wait_next_scheduler_call_args_t Struct Reference

Data Fields

- ums_context_descriptor_t **ucd**
- reason_t **reason**

3.16.1 Detailed Description

Definition at line 138 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.17 rq_yield_ums_context_args_t Struct Reference

Data Fields

- int **unused**

3.17.1 Detailed Description

Definition at line 145 of file ums_requests.h.

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

- src/UMS/common/ums_requests.h

3.18 startup_new_thread_args_t Struct Reference

Data Fields

- ums_context_descriptor_t **ucd**
- pid_t **sheduler_pid**
- void ∗(∗ **routine**)(void ∗)
- void ∗ **args_routine**

3.18.1 Detailed Description

Definition at line 46 of file ums_context.c.

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

- src/UMS/UMS/src/ums_context.c

3.19 ums_completion_list_item_t Struct Reference

element of completion_list

```
#include <ums_completion_lsit.h>
```

Data Fields

- struct list_head **list**
- int ums_context_id

3.19.1 Detailed Description

element of completion_list

Definition at line 27 of file ums_completion_lsit.h.

3.19.2 Field Documentation

3.19.2.1 ums_context_id

```
int ums_context_id
```

list field

Definition at line 29 of file ums_completion_lsit.h.

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

- src/UMS/UMS_LKM/ums_completion_lsit.h

3.20 ums_completion_list_sl_t Struct Reference

object that contains the ums_completion_list and protect it using a spin_lock

```
#include <ums_completion_lsit.h>
```

Data Fields

- int **id**
- spinlock_t ums_context_list_spin_lock
- struct list_head ums_context_list

3.20.1 Detailed Description

object that contains the ums_completion_list and protect it using a spin_lock

Definition at line 63 of file ums_completion_lsit.h.

3.20.2 Field Documentation

3.20.2.1 ums_context_list

```
struct list_head ums_context_list
```

used to protect the ums_completion_list

Definition at line 67 of file ums_completion_lsit.h.

3.20.2.2 ums_context_list_spin_lock

```
spinlock_t ums_context_list_spin_lock
```

descriptor

Definition at line 66 of file ums_completion_lsit.h.

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

- src/UMS/UMS_LKM/ums_completion_lsit.h

3.21 ums_context_sl_t Struct Reference

ums_context_SpinLock is used to protect a ums_context between several ums_schedulers

```
#include <ums_context.h>
```

Data Fields

- int **id**
- bool assigned
- spinlock_t assigned_spin_lock
- ums_context_t ∗ ums_context

3.21.1 Detailed Description

ums_context_SpinLock is used to protect a ums_context between several ums_schedulers

Definition at line 160 of file ums_context.h.

3.21.2 Field Documentation

3.21.2.1 assigned

```
bool assigned
```

descriptor, the same of the ums_context managed

Definition at line 163 of file ums_context.h.

3.21.2.2 assigned_spin_lock

```
spinlock_t assigned_spin_lock
```

(IN USE) indicates the managed ums_context has been already assigned to another scheduler

Definition at line 164 of file ums_context.h.

3.21.2.3 ums_context

```
ums_context_t* ums_context
```

used to protect "assigned" field

Definition at line 166 of file ums_context.h.

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

- src/UMS/UMS_LKM/ums_context.h

3.22 ums_context_t Struct Reference

Represents a ums_context.

```
#include <ums_context.h>
```

Data Fields

- struct list_head **list**
- struct hlist_node hlist
- pid_t pid
- int id
- void ∗ task_struct
- pid_t pid_scheduler
- struct proc_dir_entry ∗ proc_entry
- int num_switch
- int state
- void ∗(∗ routine)(void ∗args)
- void ∗ args
- void ∗ user_reserved
- u64 start_time_last_slot
- u64 ums_run_time

3.22.1 Detailed Description

Represents a ums_context.

Definition at line 32 of file ums_context.h.

3.22.2 Field Documentation

3.22.2.1 args

```
void* args
```

routine of the user

Definition at line 46 of file ums_context.h.

3.22.2.2 hlist

```
struct hlist_node hlist
```

used to arrange ums_context in ready_list

Definition at line 34 of file ums_context.h.

3.22.2.3 id

```
int id
```

thread's pid used

Definition at line 37 of file ums_context.h.

3.22.2.4 num_switch

```
int num_switch
```

entry in /proc associated to this ums_context

Definition at line 42 of file ums_context.h.

3.22.2.5 pid

```
pid_t pid
```

used by the hashtable of ums_threads, used to map thread's pid to the ums_context_descriptor

Definition at line 36 of file ums_context.h.

3.22.2.6 pid_scheduler

```
pid_t pid_scheduler
```

pointer to task_struct of thread used

Definition at line 39 of file ums_context.h.

3.22.2.7 proc_entry

```
struct proc_dir_entry* proc_entry
```

pid of the scheduler that manage the ums_context

Definition at line 41 of file ums_context.h.

3.22.2.8 routine

```
void*(* routine(void *args)
```

state of the ums_context: UMS_THREAD_STATE_IDLE, UMS_THREAD_STATE_RUNNING, UMS_THREAD_↩
STATE_ENDED

Definition at line 45 of file ums_context.h.

3.22.2.9 start_time_last_slot

```
u64 start_time_last_slot
```

user can use it as he wants, (e.g. store some characteristics of the ums_context: CPU or I/O BURST, and prio)

Definition at line 50 of file ums_context.h.

3.22.2.10 state

```
int state
```

number of switches from running to idle and viceversa

Definition at line 43 of file ums_context.h.

3.22.2.11 task_struct

```
void* task_struct
```

descriptor

Definition at line 38 of file ums_context.h.

3.22.2.12 ums_run_time

```
u64 ums_run_time
```

uses jiffies

Definition at line 51 of file ums_context.h.

3.22.2.13 user_reserved

```
void* user_reserved
```

args of user's routine

Definition at line 48 of file ums_context.h.

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

- src/UMS/UMS_LKM/ums_context.h

3.23 ums_process_t Struct Reference

Represent a ums_process object.

```
#include <ums_process.h>
```

Public Member Functions

- DECLARE_HASHTABLE (hashtable_ums_schedulers, HASHTABLE_UMS_SCHEDULERS_HASH_BITS)
- DECLARE_HASHTABLE (hashtable_ums_threads, HASHTABLE_UMS_THREADS_HASH_BITS)

Data Fields

- struct hlist_node **hlist**
- int key
- rwlock_t hashtable_ums_schedulers_rwlock
- rwlock_t hashtable_ums_threads_rwlock
- struct idr idr_completion_list
- rwlock_t idr_completion_list_rwlock
- struct idr idr_ums_context
- rwlock_t idr_ums_context_rwlock
- struct proc_dir_entry ∗ proc_entry
- struct proc_dir_entry ∗ proc_entry_main_scheds

3.23.1 Detailed Description

Represent a ums_process object.

Definition at line 23 of file ums_process.h.

3.23.2 Member Function Documentation

3.23.2.1 DECLARE_HASHTABLE() [1/2]

```
DECLARE_HASHTABLE (
             hashtable_ums_schedulers ,
             HASHTABLE_UMS_SCHEDULERS_HASH_BITS  )
```

key in the ums_hashtable (equals to tgid (thread id))

3.23.2.2 DECLARE_HASHTABLE() [2/2]

```
DECLARE_HASHTABLE (
             hashtable_ums_threads ,
             HASHTABLE_UMS_THREADS_HASH_BITS  )
```

rw_spin_lock of ums_scheduler_hashtable

3.23.3 Field Documentation

3.23.3.1 hashtable_ums_schedulers_rwlock

```
rwlock_t hashtable_ums_schedulers_rwlock
```

hashtable that contains schedulers, the key of as scheduler is its pid

Definition at line 28 of file ums_process.h.

3.23.3.2 hashtable_ums_threads_rwlock

```
rwlock_t hashtable_ums_threads_rwlock
```

hashtable used to map a thread to its ums_context

Definition at line 31 of file ums_process.h.

3.23.3.3 idr_completion_list

```
struct idr idr_completion_list
```

rw_spin_lock of the ums_thraed_hashtable

Definition at line 34 of file ums_process.h.

3.23.3.4 idr_completion_list_rwlock

```
rwlock_t idr_completion_list_rwlock
```

idr struct used to allocate the ums_completion_lists managed by this process

Definition at line 35 of file ums_process.h.

3.23.3.5 idr_ums_context

```
struct idr idr_ums_context
```

rw_spin_lock of idr_completion_list

Definition at line 37 of file ums_process.h.

3.23.3.6 idr_ums_context_rwlock

```
rwlock_t idr_ums_context_rwlock
```

idr struct used to store ums_contexts managed by this process

Definition at line 38 of file ums_process.h.

3.23.3.7 key

```
int key
```

field used to arrange it in the ums_hashmap

Definition at line 25 of file ums_process.h.

3.23.3.8 proc_entry

```
struct proc_dir_entry* proc_entry
```

rw_spin_lock of idr_ums_context

Definition at line 40 of file ums_process.h.

3.23.3.9 proc_entry_main_scheds

```
struct proc_dir_entry* proc_entry_main_scheds
```

entry in /proc, corresponds to /proc/ums/<tgid>

Definition at line 41 of file ums_process.h.

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

- src/UMS/UMS_LKM/ums_process.h

3.24 ums_scheduler_sl_t Struct Reference

object used to arrange a ums_scheduler in a hashtable and to protect it with a spin_lock

```
#include <ums_scheduler.h>
```

Data Fields

- struct hlist_node **hlist**
- int key
- spinlock_t ums_scheduler_spin_lock
- ums_scheduler_t ∗ ums_scheduler
- struct proc_dir_entry ∗ proc_entry
- struct proc_dir_entry ∗ proc_entry_info
- struct proc_dir_entry ∗ proc_entry_main_workers

3.24.1 Detailed Description

object used to arrange a ums_scheduler in a hashtable and to protect it with a spin_lock

Definition at line 255 of file ums_scheduler.h.

3.24.2 Field Documentation

3.24.2.1 key

```
int key
```

used to arrange in the hashtable of process' schedulers

Definition at line 257 of file ums_scheduler.h.

3.24.2.2 proc_entry

```
struct proc_dir_entry* proc_entry
```

pointer to the scheduler to protect

Definition at line 262 of file ums_scheduler.h.

3.24.2.3 proc_entry_info

```
struct proc_dir_entry* proc_entry_info
```

entry in /proc, corresponds to /proc/ums/<tgid>/schedulers/<pid>

Definition at line 263 of file ums_scheduler.h.

3.24.2.4 proc_entry_main_workers

```
struct proc_dir_entry* proc_entry_main_workers
```

entry in /proc, corresponds to /proc/ums/<tgid>/schedulers/<pid>/info

Definition at line 264 of file ums_scheduler.h.

3.24.2.5 ums_scheduler

```
ums_scheduler_t* ums_scheduler
```

protect ums_scheduler

Definition at line 260 of file ums_scheduler.h.

3.24.2.6 ums_scheduler_spin_lock

```
spinlock_t ums_scheduler_spin_lock
```

key in the hashtable, corresponds to scheduler's pid

Definition at line 259 of file ums_scheduler.h.

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

- src/UMS/UMS_LKM/ums_scheduler.h

3.25 ums_scheduler_t Struct Reference

object that represent a ums_scheduler

```
#include <ums_scheduler.h>
```

Data Fields

- void ∗ **scheduler_task_struct**
- ums_completion_list_sl_t ∗ completion_list
- struct list_head ∗ current_completion_list_item
- struct list_head ready_list
- struct list_head ∗ current_ready_list_item
- ums_context_t ∗ running_thread
- entry_point_args_t ∗ entry_point_args
- int num_switch
- int cpu_core

3.25.1 Detailed Description

object that represent a ums_scheduler

Definition at line 37 of file ums_scheduler.h.

3.25.2 Field Documentation

3.25.2.1 completion_list

```
ums_completion_list_sl_t* completion_list
```

task_struct of the scheduler thread

Definition at line 40 of file ums_scheduler.h.

3.25.2.2 cpu_core

```
int cpu_core
```

number of scheduler calls

Definition at line 52 of file ums_scheduler.h.

3.25.2.3 current_completion_list_item

```
struct list_head* current_completion_list_item
```

ums_completion_list managed

Definition at line 41 of file ums_scheduler.h.

3.25.2.4 current_ready_list_item

```
struct list_head* current_ready_list_item
```

ready list of the scheduler

Definition at line 44 of file ums_scheduler.h.

3.25.2.5 entry_point_args

```
entry_point_args_t* entry_point_args
```

pointer to the current ums_context in execution

Definition at line 48 of file ums_scheduler.h.

3.25.2.6 num_switch

```
int num_switch
```

args of the entry_point function of the scheduler

Definition at line 50 of file ums_scheduler.h.

3.25.2.7 ready_list

```
struct list_head ready_list
```

current ums_completion_list_item during navigation of the ums_completion_list

Definition at line 43 of file ums_scheduler.h.

3.25.2.8 running_thread

```
ums_context_t* running_thread
```

current ums_context during navigation of ready_list

Definition at line 46 of file ums_scheduler.h.

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

- src/UMS/UMS_LKM/ums_scheduler.h

Chapter 4

File Documentation

4.1 src/UMS/common/ums_requests.h File Reference

```
#include "ums_types.h"
```

Data Structures

- struct rq_create_delete_process_args_t
- struct rq_create_delete_completion_list_args_t
- struct rq_create_delete_ums_context_args_t
- struct rq_completion_list_add_remove_ums_context_args_t
- struct rq_create_delete_ums_scheduler_args_t
- struct rq_execute_next_new_thread_args_t
- struct rq_execute_next_ready_thread_args_t
- struct rq_startup_new_thread_args_t
- struct rq_end_thread_args_t
- struct rq_wait_next_scheduler_call_args_t
- struct rq_yield_ums_context_args_t
- struct rq_get_from_cl_args_t
- struct rq_get_from_rl_args_t
- struct rq_execute_args_t

Macros

- #define **REQUEST_0** 120
- #define **REQUEST_1** 119
- #define **REQUEST_2** 118
- #define **REQUEST_3** 117
- #define **REQUEST_4** 116
- #define **REQUEST_5** 115
- #define **REQUEST_6** 114
- #define **REQUEST_7** 113
- #define **REQUEST_8** 112
- #define **REQUEST_9** 111
- #define **REQUEST_10** 110

- #define **REQUEST_11** 109
- #define **REQUEST_12** 108
- #define **REQUEST_13** 107
- #define **REQUEST_14** 106
- #define **REQUEST_15** 105
- #define **REQUEST_16** 104
- #define **REQUEST_17** 103
- #define **REQUEST_18** 102
- #define **REQUEST_19** 101
- #define **REQUEST_20** 100
- #define **REQUEST_DEBUG_0** 255
- #define **REQUEST_DEBUG_1** 254
- #define **REQUEST_DEBUG_2** 253
- #define **REQUEST_DEBUG_3** 252
- #define **REQUEST_DEBUG_4** 251
- #define **RQ_CREATE_PROCESS** REQUEST_0
- #define **RQ_DELETE_PROCESS** REQUEST_1
- #define **RQ_CREATE_COMPLETION_LIST** REQUEST_2
- #define **RQ_DELETE_COMPLETION_LIST** REQUEST_3
- #define **RQ_CREATE_UMS_CONTEXT** REQUEST_4
- #define **RQ_DELETE_UMS_CONTEXT** REQUEST_5
- #define **RQ_COMPLETION_LIST_ADD_UMS_CONTEXT** REQUEST_6
- #define **RQ_COMPLETION_LIST_REMOVE_UMS_CONTEXT** REQUEST_7
- #define **RQ_CREATE_UMS_SCHEDULER** REQUEST_8
- #define **RQ_EXIT_UMS_SCHEDULER** REQUEST_9
- #define **RQ_EXECUTE_NEXT_NEW_THREAD** REQUEST_10
- #define **RQ_EXECUTE_NEXT_READY_THREAD** REQUEST_11
- #define **RQ_STARTUP_NEW_THREAD** REQUEST_12
- #define **RQ_END_THREAD** REQUEST_13
- #define **RQ_WAIT_NEXT_SCHEDULER_CALL** REQUEST_14
- #define **RQ_YIELD_UMS_CONTEXT** REQUEST_15
- #define **RQ_GET_FROM_CL** REQUEST_16
- #define **RQ_GET_FROM_RL** REQUEST_17
- #define **RQ_EXECUTE** REQUEST_18
- #define **RQ_EXECUTE_READY_LIST** REQUEST_19

Typedefs

- typedef struct rq_create_delete_process_args_t **rq_create_delete_process_args_t**
- typedef struct rq_create_delete_completion_list_args_t **rq_create_delete_completion_list_args_t**
- typedef struct rq_create_delete_ums_context_args_t **rq_create_delete_ums_context_args_t**
- typedef struct rq_completion_list_add_remove_ums_context_args_t **rq_completion_list_add_remove_↩ ums_context_args_t**
- typedef struct rq_create_delete_ums_scheduler_args_t **rq_create_delete_ums_scheduler_args_t**
- typedef struct rq_execute_next_new_thread_args_t **rq_execute_next_new_thread_args_t**
- typedef struct rq_execute_next_ready_thread_args_t **rq_execute_next_ready_thread_args_t**
- typedef struct rq_startup_new_thread_args_t **rq_startup_new_thread_args_t**
- typedef struct rq_end_thread_args_t **rq_end_thread_args_t**
- typedef struct rq_wait_next_scheduler_call_args_t **rq_wait_next_scheduler_call_args_t**
- typedef struct rq_yield_ums_context_args_t **rq_yield_ums_context_args_t**
- typedef struct rq_get_from_cl_args_t **rq_get_from_cl_args_t**
- typedef struct rq_get_from_rl_args_t **rq_get_from_rl_args_t**
- typedef struct rq_execute_args_t **rq_execute_args_t**

4.1.1 Detailed Description

This file contains all the request macros

4.2 src/UMS/common/ums_types.h File Reference

Data Structures

- struct entry_point_args_t

    arguments of a entry_point function
- struct info_ums_context_t

    used to choose a ums_context from the ready list or from the completion_list

Macros

- #define **REASON_0** 100
- #define **REASON_1** 101
- #define **REASON_2** 102
- #define **REASON_3** 103
- #define **REASON_4** 104
- #define **REASON_5** 105
- #define **REASON_6** 106
- #define **REASON_7** 107
- #define **REASON_SPECIAL_0** 200
- #define **REASON_SPECIAL_1** 201
- #define **REASON_SPECIAL_2** 202
- #define **REASON_SPECIAL_3** 203
- #define **RES_ERR_0** 300
- #define **RES_ERR_1** 301
- #define **RES_ERR_2** 302
- #define **RES_ERR_3** 303
- #define **RES_ERR_4** 304
- #define **RES_ERR_5** 305
- #define **RES_ERR_6** 306
- #define **RES_ERR_7** 307
- #define **SUCCESS** 0
- #define **FAILURE** -1
- #define **ERR_EMPTY_COMP_LIST** RES_ERR_0
- #define **ERR_EMPTY_READY_LIST** RES_ERR_1
- #define **ERR_INVALID_CLD** RES_ERR_2
- #define **ERR_INVALID_UCD** RES_ERR_3
- #define **ERR_INTERNAL** RES_ERR_4 /*SHOULD BE A KERNEL PANIC*/
- #define **ERR_ASSIGNED** RES_ERR_5
- #define **ERR_CPU_SELECTED** RES_ERR_6
- #define **REASON_STARTUP** REASON_0
- #define **REASON_THREAD_BLOCKED** REASON_1
- #define **REASON_THREAD_YIELD** REASON_2
- #define **REASON_THREAD_ENDED** REASON_3
- #define **REASON_SPECIAL_END_SCHEDULER** REASON_SPECIAL_0

Typedefs

- typedef int **ums_context_descriptor_t**
- typedef int **ums_completion_list_descriptor_t**
- typedef int **res_t**
- typedef int **reason_t**
- typedef struct entry_point_args_t entry_point_args_t

  arguments of a entry_point function
- typedef struct info_ums_context_t info_ums_context_t

  used to choose a ums_context from the ready list or from the completion_list

4.2.1 Detailed Description

This file contains all variable types that must be shared by user and kerenl module

4.2.2 Typedef Documentation

4.2.2.1 entry_point_args_t

```
typedef struct entry_point_args_t entry_point_args_t
```

arguments of a entry_point function

4.2.2.2 info_ums_context_t

```
typedef struct info_ums_context_t info_ums_context_t
```

used to choose a ums_context from the ready list or from the completion_list

4.3 src/UMS/UMS/src/ums.h File Reference

```
#include <stdbool.h>
#include <unistd.h>
#include <stdint.h>
#include <pthread.h>
#include "../../common/ums_requests.h"
```

Typedefs

- typedef pthread_t **ums_scheduler_descriptor_t**

Functions

- res_t ums_init (void)
- res_t ums_destroy (void)
- res_t create_ums_context (ums_context_descriptor_t ∗descriptor, void ∗(∗routine)(void ∗), void ∗args, void ∗user_res)
- res_t delete_ums_context (ums_context_descriptor_t descriptor)
- res_t create_ums_completion_list (ums_completion_list_descriptor_t ∗ums_completion_list_descriptor)

  Create a ums completion list object.
- res_t delete_ums_completion_list (ums_completion_list_descriptor_t ums_completion_list_descriptor)

  Delete a ums completion list object.
- res_t completion_list_add_ums_context (ums_completion_list_descriptor_t completion_list_d, ums_↩ context_descriptor_t ums_context_d)

  Add a ums_context to a ums_completion_list.
- res_t completion_list_remove_ums_context (ums_completion_list_descriptor_t completion_list_d, ums_↩ context_descriptor_t ums_context_d)

  Remove a ums_context from a ums_completion_list.
- res_t create_ums_scheduler (ums_scheduler_descriptor_t ∗sd, ums_completion_list_descriptor_t cd, void(∗entry_point)(entry_point_args_t ∗entry_point_args), void ∗sched_args, int cpu_core)

  Create a ums scheduler object.
- void exit_scheduler (int return_value)

  exit() function for the scheduler
- res_t execute_next_new_thread (void)

  Execute the next ums_context in the ums_completion_list of the scheduler.
- res_t execute_next_ready_thread (void)

  Execute the next ums_context in the ready_list of the scheduler.
- res_t join_scheduler (ums_scheduler_descriptor_t ∗usd, int ∗return_value)

  Join scheduler thread.
- res_t yield (void)

  Current ums_context in execution leaves the control to the scheduler.
- res_t get_ums_contexts_from_cl (info_ums_context_t ∗array_info_ums_context, size_t array_size)

  Get the ums contexts from the completion_list of the scheduler.
- res_t get_ums_contexts_from_rl (info_ums_context_t ∗array_info_ums_context, size_t array_size)

  Get the ums contexts from the ready_list of the scheduler.
- res_t execute (info_ums_context_t ∗info_ums_context)

  execute a ums_context identified by a pointer to a info_ums_context_t

Variables

- pid_t **tgid**
- int **ums_fd**

4.3.1 Detailed Description

This file contains all functions available to the user

4.3.2 Function Documentation

4.3.2.1 completion_list_add_ums_context()

```
res_t completion_list_add_ums_context (
            ums_completion_list_descriptor_t completion_list_d,
            ums_context_descriptor_t ums_context_d )
```

Add a ums_context to a ums_completion_list.

It performs a RQ_COMPLETION_LIST_ADD_UMS_CONTEXT request

Parameters

completion_list↩ _d	Descriptor of the ums_completion_list
ums_context_d	Descriptor of the ums_context to add

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 31 of file ums_completion_list.c.

4.3.2.2 completion_list_remove_ums_context()

```
res_t completion_list_remove_ums_context (
            ums_completion_list_descriptor_t completion_list_d,
            ums_context_descriptor_t ums_context_d )
```

Remove a ums_context from a ums_completion_list.

It performs a RQ_COMPLETION_LIST_REMOVE_UMS_CONTEXT request

Parameters

completion_list↩ _d	Descriptor of the ums_completion_list
ums_context_d	Descriptor of the ums_context to remove

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 39 of file ums_completion_list.c.

4.3.2.3 create_ums_completion_list()

```
res_t create_ums_completion_list (
            ums_completion_list_descriptor_t * ums_completion_list_descriptor )
```

Create a ums completion list object.

It performs a RQ_CREATE_COMPLETION_LIST request

Parameters

ums_completion_list_descriptor	Pointer to where to store the descriptor of the new ums_completions_list

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 13 of file ums_completion_list.c.

4.3.2.4 create_ums_context()

```
res_t create_ums_context (
            ums_context_descriptor_t * descriptor,
            void *(*)(void *) routine,
            void * args,
            void * user_res )
```

Creates a ums_context object

It performs a RQ_CREATE_UMS_CONTEXT request

Parameters

descriptor	Pointer used to save the ums_context_descriptor assigned
routine	Function poiter to the routine of the new ums_context
args	Arguments to be passed to the ums_context's routine
user_res	user managed object

Returns

Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 19 of file ums_context.c.

4.3.2.5 create_ums_scheduler()

```
res_t create_ums_scheduler (
            ums_scheduler_descriptor_t * sd,
            ums_completion_list_descriptor_t cd,
            void(*)(entry_point_args_t *entry_point_args) entry_point,
            void * sched_args,
            int cpu_core )
```

Create a ums scheduler object.

It creates a new thread that will be used as scheduler. The new thread performs a RQ_CREATE_UMS_SCHED↩
ULER request NOTE: entry_point must have a specific structure composed by a swicth-case statement

Parameters

sd	Pointer used to store the descriptor of the new ums_scheduler
cd	Descriptor of the ums_completion_list to use
entry_point	Entry_point function of the scheduler
sched_args	Arguments to pass to entry_point functions
cpu_core	CPU core to use

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 76 of file ums_scheduler.c.

4.3.2.6 delete_ums_completion_list()

```
res_t delete_ums_completion_list (
            ums_completion_list_descriptor_t ums_completion_list_descriptor )
```

Delete a ums completion list object.

It performs a RQ_REMOVE_COMPLETION_LIST request

Parameters

ums_completion_list_descriptor	Descriptor of the ums_completion_list to delete

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 21 of file ums_completion_list.c.

4.3.2.7 delete_ums_context()

```
res_t delete_ums_context (
            ums_context_descriptor_t descriptor )
```

Deletes a ums_context

Deletes the ums_context indicated by descriptor parameter, it performs a RQ_DELETE_UMS_CONTEXT request

Parameters

descriptor	Descriptor of the ums_context to delete

Returns

Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 32 of file ums_context.c.

4.3.2.8 execute()

```
res_t execute (
            info_ums_context_t * info_ums_context )
```

execute a ums_context identified by a pointer to a info_ums_context_t

Parameters

info_ums_context	pointer to the info_ums_context_t object to execute

Returns

return 0 on success, return -1 on failure and set errno according to

Definition at line 149 of file ums_context.c.

4.3.2.9 execute_next_new_thread()

```
res_t execute_next_new_thread (
            void )
```

Execute the next ums_context in the ums_completion_list of the scheduler.

It performs a RQ_EXECUTE_NEXT_NEW_THREAD request to get the routine and the arguments of the ums↩ _context then it creates a new threads NOTE: It always returns a value, due to the fact that at every call of the entry_point of the scheduler, the entire entry_point function is executed!

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 83 of file ums_context.c.

4.3.2.10 execute_next_ready_thread()

```
res_t execute_next_ready_thread (
            void  )
```

Execute the next ums_context in the ready_list of the scheduler.

It performs a RQ_EXECUTE_NEXT_READY_THREAD request NOTE: It always returns a value, due to the fact that at every call of the entry_point of the scheduler, the entire entry_point function is executed!

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 131 of file ums_context.c.

4.3.2.11 exit_scheduler()

```
void exit_scheduler (
            int return_value )
```

exit() function for the scheduler

This function must be used in the entry_point function and replaces the classic exit() syscall. It performs a RQ_E↩XIT_SCHEDULER request NOTE: The user MUST use this function in order to terminate the scheduler's thread

Parameters

return_value	Exit value of the scheduler thread

Definition at line 112 of file ums_scheduler.c.

4.3.2.12 get_ums_contexts_from_cl()

```
res_t get_ums_contexts_from_cl (
            info_ums_context_t * array_info_ums_context,
            size_t array_size )
```

Get the ums contexts from the completion_list of the scheduler.

Parameters

array_info_ums_context	output, array of info_ums_context_t
array_size	input, size of array, maximum number of ums_context to read

Returns

return (>0) the number of context readed, return -1 on failure and set errno according to

NOTE: After a successfully execution, execute() function must be always called

Definition at line 128 of file ums_scheduler.c.

4.3.2.13 get_ums_contexts_from_rl()

```
res_t get_ums_contexts_from_rl (
            info_ums_context_t * array_info_ums_context,
            size_t array_size )
```

Get the ums contexts from the ready_list of the scheduler.

Parameters

array_info_ums_context	output, array of info_ums_context_t
array_size	input, size of array, maximum number of ums_context to read

Returns

return (>0) the number of context readed, return -1 on failure and set errno according to

NOTE: After a successfully execution, execute() function must be always called

Definition at line 138 of file ums_scheduler.c.

4.3.2.14 join_scheduler()

```
res_t join_scheduler (
            ums_scheduler_descriptor_t * usd,
            int * return_value )
```

Join scheduler thread.

It waits the end of the scheduler thread indicated by the ums_scheduler_descriptor pointer

Parameters

usd	Poiter to the ums_scheduler_descriptor
return_value	Pointer to where store the return value of the scheduler thread

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 18 of file ums_scheduler.c.

4.3.2.15 ums_destroy()

```
res_t ums_destroy (
            void  )
```

Destroys UMS

Closes the UMS virtual device and destorys the ums_process entity, it performs a RQ_DELETE_PROCESS request

Returns

Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 37 of file ums.c.

4.3.2.16 ums_init()

```
res_t ums_init (
            void  )
```

Initializes UMS

Opens the UMS virtual device and creates a ums_process entity, it performs a RQ_CREATE_PROCESS request

Returns

Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 21 of file ums.c.

4.3.2.17 yield()

```
res_t yield (
            void  )
```

Current ums_context in execution leaves the control to the scheduler.

It performs a RQ_YIELD_UMS_CONTEXT request

Returns

res_t Returns 0 on sucess, otherwise -1 and sets errno according to

Definition at line 141 of file ums_context.c.

4.4 src/UMS/UMS_LKM/rq_ums_completion_list.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/miscdevice.h>
#include <linux/sched.h>
#include <asm/uaccess.h>
#include <linux/proc_fs.h>
#include "../common/ums_requests.h"
#include "../common/ums_types.h"
#include "ums_hashtable.h"
```

4.4.1 Detailed Description

This file contains all the requests used to manage a ums_completion_list

4.5 src/UMS/UMS_LKM/rq_ums_context.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/miscdevice.h>
#include <linux/sched.h>
#include <asm/uaccess.h>
#include <linux/proc_fs.h>
#include "../common/ums_requests.h"
#include "../common/ums_types.h"
```

4.5.1 Detailed Description

This file contains all the requests used to manage a ums_context

4.6 src/UMS/UMS_LKM/rq_ums_process.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/miscdevice.h>
#include <linux/sched.h>
#include <asm/uaccess.h>
#include <linux/proc_fs.h>
#include "../common/ums_requests.h"
#include "../common/ums_types.h"
#include "ums_hashtable.h"
```

4.6.1 Detailed Description

This file contains all the requests used to manage a ums_process

4.7 src/UMS/UMS_LKM/rq_ums_scheduler.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/miscdevice.h>
#include <linux/sched.h>
#include <asm/uaccess.h>
#include <linux/proc_fs.h>
#include "../common/ums_requests.h"
#include "../common/ums_types.h"
#include "ums_hashtable.h"
#include "ums_proc.h"
```

4.7.1 Detailed Description

This file contains all the requests used to manage a ums_scheduler

4.8 src/UMS/UMS_LKM/ums_completion_lsit.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/idr.h>
#include <linux/hashtable.h>
#include <linux/spinlock.h>
#include <stdbool.h>
#include <linux/list.h>
#include <linux/rwlock.h>
#include "../common/ums_types.h"
#include "ums_context.h"
```

Data Structures

- struct ums_completion_list_item_t

  element of completion_list
- struct ums_completion_list_sl_t

  object that contains the ums_completion_list and protect it using a spin_lock

Macros

- #define INIT_UMS_COMPLETION_LIST_ITEM(p_ums_completion_list_item, ums_context_id_in)

  constructor ums_completion_list_item
- #define DESTROY_UMS_COMPLETION_LIST_ITEM(p_ums_completion_list_item)

  destructor ums_completion_list_item
- #define INIT_UMS_COMPLETION_LIST_SL(p_ums_completion_list_sl)

  constructor ums_completion_list_sl object
- #define DESTROY_UMS_COMPLETION_LIST_SL(p_ums_completion_list_sl)

  destructor ums_completion_list_sl object
- #define ums_completion_list_add_item(p_ums_completion_list, p_ums_completion_list_item)

  add a ums_completion_list item to the ums_completion_list
- #define ums_completion_list_remove_item(p_ums_completion_list, p_ums_completion_list_item)

  remove a ums_completion_list item to the ums_completion_list
- #define ums_completion_list_sl_lock_get_list(p_ums_completion_list_sl, p_list_head)

  get and lock the ums_completion_list
- #define ums_completion_list_sl_unlock_list(p_ums_completion_list_sl)

  unlock the ums_completion_list
- #define ums_completion_list_remove_item_by_descriptor(p_ums_completion_list, ums_context_descriptor, p_ums_completion_list_item)

  remove a ums_completion_list_item by the ums_context descriptor to which it refers to
- #define ums_completion_list_remove_item_by_descriptor_no_sl(p_ums_completion_list, ums_context_↩ descriptor, p_ums_completion_list_item)

  without ue of spin_lock, remove a ums_completion_list_item by the ums_context descriptor to which it refers to
- #define ums_completion_list_remove_first(p_ums_completion_list, p_ums_completion_list_item_OUT)

  remove first element from the ums_completion_list
- #define PRINTK_UMS_COMPLETION_LIST_ITEM(p_obj, PREFIX)

  printK ums_completion_list_item
- #define PRINTK_UMS_COMPLETION_LIST_SL(p_obj, PREFIX)

  printK ums_completion_list_sl object

Typedefs

- typedef struct ums_completion_list_item_t ums_completion_list_item_t

  element of completion_list
- typedef struct ums_completion_list_sl_t ums_completion_list_sl_t

  object that contains the ums_completion_list and protect it using a spin_lock

4.8.1 Detailed Description

This file contains definitions and functions of objects related to a ums_completion_list

4.8.2 Macro Definition Documentation

4.8.2.1 DESTROY_UMS_COMPLETION_LIST_ITEM

```
#define DESTROY_UMS_COMPLETION_LIST_ITEM(
              p_ums_completion_list_item )
```

Value:

```
    do{ \
        (p_ums_completion_list_item)->ums_context_id = -1;   \
    }while(0)
```

destructor ums_completion_list_item

Parameters

p_ums_completion_list_item	object to destroy

Definition at line 51 of file ums_completion_lsit.h.

4.8.2.2 DESTROY_UMS_COMPLETION_LIST_SL

```
#define DESTROY_UMS_COMPLETION_LIST_SL(
              p_ums_completion_list_sl )
```

Value:

```
    do{ \
        (p_ums_completion_list_sl)->id = -1; \
    }while(0)
```

destructor ums_completion_list_sl object

Parameters

p_ums_completion_list↩ _sl	object to destroy

Definition at line 89 of file ums_completion_lsit.h.

4.8.2.3 INIT_UMS_COMPLETION_LIST_ITEM

```
#define INIT_UMS_COMPLETION_LIST_ITEM(
              p_ums_completion_list_item,
              ums_context_id_in )
```

Value:
```
    do{ \
        (p_ums_completion_list_item)->ums_context_id = ums_context_id_in;    \
    }while(0)
```

constructor ums_completion_list_item

Parameters

p_ums_completion_list_item	object to init
ums_context_id_in	id of the referred ums_context

Definition at line 40 of file ums_completion_lsit.h.

4.8.2.4 INIT_UMS_COMPLETION_LIST_SL

```
#define INIT_UMS_COMPLETION_LIST_SL(
                p_ums_completion_list_sl )
```

Value:
```
    do{ \
        (p_ums_completion_list_sl)->id = -1; \
        \
        spin_lock_init(&(p_ums_completion_list_sl)->ums_context_list_spin_lock);   \
        INIT_LIST_HEAD(&(p_ums_completion_list_sl)->ums_context_list); \
    }while(0)
```

constructor ums_completion_list_sl object

Parameters

p_ums_completion_list↩ _sl	object to init

Definition at line 76 of file ums_completion_lsit.h.

4.8.2.5 PRINTK_UMS_COMPLETION_LIST_ITEM

```
#define PRINTK_UMS_COMPLETION_LIST_ITEM(
                p_obj,
                PREFIX )
```

Value:
```
    do{ \
        char* __buff = kmalloc(512, GFP_KERNEL);     \
        if( snprintf_ums_completion_list_item(__buff, 512, p_obj) > 512-4)    \
            printk(KERN_DEBUG "\n PRINTK_UMS_COMPLETION_LIST_ITEM() overflow!!!\n");     \
        else printk(PREFIX "ums_completion_list_item_t = \n{\n%s}\n", __buff); \
        kfree(__buff);  \
    }while(0)
```

printK ums_completion_list_item

Definition at line 233 of file ums_completion_lsit.h.

4.8.2.6 PRINTK_UMS_COMPLETION_LIST_SL

```
#define PRINTK_UMS_COMPLETION_LIST_SL(
              p_obj,
              PREFIX )
```

Value:
```
    do{ \
        char* __buff = kmalloc(4096, GFP_KERNEL);   \
        if( snprintf_ums_completion_list_sl(__buff, 4096, p_obj) > 4096-4)  \
            printk(KERN_DEBUG "\n PRINTK_UMS_COMPLETION_LIST_SL() overflow!!!\n");   \
        else printk(PREFIX "\tums_completion_list_sl_t = \n\t\t{\n%s\t\t}\n", __buff);    \
        kfree(__buff);  \
    }while(0)
```

printK ums_completion_list_sl object

Definition at line 301 of file ums_completion_lsit.h.

4.8.2.7 ums_completion_list_add_item

```
#define ums_completion_list_add_item(
              p_ums_completion_list,
              p_ums_completion_list_item )
```

Value:
```
    do{ \
        spin_lock(&((p_ums_completion_list)->ums_context_list_spin_lock));  \
            list_add_tail(&((p_ums_completion_list_item)->list),
       &((p_ums_completion_list)->ums_context_list));  \
        spin_unlock(&((p_ums_completion_list)->ums_context_list_spin_lock));  \
    }while(0)
```

add a ums_completion_list item to the ums_completion_list

Parameters

p_ums_completion_list	pointer to ums_context_list (ums_completion_list)
p_ums_completion_list_item	pointer to the element to add

Definition at line 103 of file ums_completion_lsit.h.

4.8.2.8 ums_completion_list_remove_first

```
#define ums_completion_list_remove_first(
              p_ums_completion_list,
              p_ums_completion_list_item_OUT )
```

Value:
```
    do{ \
        spin_lock(&((p_ums_completion_list)->ums_context_list_spin_lock));  \
        p_ums_completion_list_item_OUT =
       list_first_entry_or_null(&(p_ums_completion_list)->ums_context_list, ums_completion_list_item_t,
       list);    \
        if(likely((p_ums_completion_list_item_OUT) != NULL))    \
            list_del(&((p_ums_completion_list_item_OUT)->list));   \
        spin_unlock(&((p_ums_completion_list)->ums_context_list_spin_lock));   \
    }while(0)
```

remove first element from the ums_completion_list

Parameters

p_ums_completion_list	pointer to ums_completion_list_sl
p_ums_completion_list_item_OUT,output,removed	element

NOTE: If the list is empty, it return NULL

Definition at line 210 of file ums_completion_lsit.h.

4.8.2.9 ums_completion_list_remove_item

```
#define ums_completion_list_remove_item(
              p_ums_completion_list,
              p_ums_completion_list_item )
```

Value:
```
    do{ \
        spin_lock(&((p_ums_completion_list)->ums_context_list_spin_lock));  \
            list_del(&((p_ums_completion_list_item)->list));  \
        spin_unlock(&((p_ums_completion_list)->ums_context_list_spin_lock));   \
    }while(0)
```

remove a ums_completion_list item to the ums_completion_list

Parameters

p_ums_completion_list	pointer to ums_context_list (ums_completion_list)
p_ums_completion_list_item	pointer to the element to remove

Definition at line 117 of file ums_completion_lsit.h.

4.8.2.10 ums_completion_list_remove_item_by_descriptor

```
#define ums_completion_list_remove_item_by_descriptor(
              p_ums_completion_list,
              ums_context_descriptor,
              p_ums_completion_list_item )
```

Value:
```
    do{ \
        struct list_head* current_item_list;    \
        ums_completion_list_item_t* current_item;    \
        \
        spin_lock(&((p_ums_completion_list)->ums_context_list_spin_lock));  \
        list_for_each(current_item_list, &((p_ums_completion_list)->ums_context_list)){ \
            current_item = list_entry(current_item_list, ums_completion_list_item_t, list); \
            if(unlikely(current_item->ums_context_id == ums_context_descriptor)){   \
                list_del(&((current_item)->list));  \
                p_ums_completion_list_item = current_item; \
                break;  \
            }   \
        }   \
        spin_unlock(&((p_ums_completion_list)->ums_context_list_spin_lock));  \
    }while(0)
```

remove a ums_completion_list_item by the ums_context descriptor to which it refers to

Parameters

p_ums_completion_list	pointer to ums_completion_list_sl
ums_context_descriptor	descriptor of the ums_context
p_ums_completion_list_item	output, ums_completion_list_item removed

Definition at line 159 of file ums_completion_lsit.h.

4.8.2.11 ums_completion_list_remove_item_by_descriptor_no_sl

```
#define ums_completion_list_remove_item_by_descriptor_no_sl(
              p_ums_completion_list,
              ums_context_descriptor,
              p_ums_completion_list_item )
```

Value:

```
    do{ \
        struct list_head* current_item_list;    \
        ums_completion_list_item_t* current_item;   \
        \
        list_for_each(current_item_list, &((p_ums_completion_list)->ums_context_list)){ \
            current_item = list_entry(current_item_list, ums_completion_list_item_t, list); \
            if(unlikely(current_item->ums_context_id == ums_context_descriptor)){   \
                list_del(&((current_item)->list));  \
                p_ums_completion_list_item = current_item; \
                break;  \
            }   \
        }   \
    }while(0)
```

without ue of spin_lock, remove a ums_completion_list_item by the ums_context descriptor to which it refers to

Parameters

p_ums_completion_list	pointer to ums_completion_list_sl
ums_context_descriptor	descriptor of the ums_context
p_ums_completion_list_item	output, ums_completion_list_item removed

NOTE: This function assumes that spin_lock has been already called

Definition at line 185 of file ums_completion_lsit.h.

4.8.2.12 ums_completion_list_sl_lock_get_list

```
#define ums_completion_list_sl_lock_get_list(
             p_ums_completion_list_sl,
             p_list_head )
```

Value:
```
    do{ \
        spin_lock(&((p_ums_completion_list_sl)->ums_context_list_spin_lock));  \
        p_list_head = &((p_ums_completion_list_sl)->ums_context_list);  \
    }while(0)
```

get and lock the ums_completion_list

Parameters

p_ums_completion_list←_sl	pointer to ums_completion_list_sl
p_list_head	pointer to the actual ums_completion_list

Definition at line 133 of file ums_completion_lsit.h.

4.8.2.13 ums_completion_list_sl_unlock_list

```
#define ums_completion_list_sl_unlock_list(
             p_ums_completion_list_sl )
```

Value:
```
    do{ \
        spin_unlock(&((p_ums_completion_list_sl)->ums_context_list_spin_lock));  \
    }while(0)
```

unlock the ums_completion_list

Parameters

p_ums_completion_list←_sl	pointer to ums_context_list_sl

Definition at line 145 of file ums_completion_lsit.h.

4.8.3 Typedef Documentation

4.8.3.1 ums_completion_list_item_t

```
typedef struct ums_completion_list_item_t ums_completion_list_item_t
```

element of completion_list

4.8.3.2 ums_completion_list_sl_t

```
typedef struct ums_completion_list_sl_t ums_completion_list_sl_t
```

object that contains the ums_completion_list and protect it using a spin_lock

4.9 src/UMS/UMS_LKM/ums_context.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/idr.h>
#include <linux/hashtable.h>
#include <linux/spinlock.h>
#include <stdbool.h>
#include <linux/list.h>
#include <linux/rwlock.h>
#include "../common/ums_types.h"
#include <linux/proc_fs.h>
#include <linux/jiffies.h>
```

Data Structures

- struct ums_context_t

    Represents a ums_context.
- struct ums_context_sl_t

    ums_context_SpinLock is used to protect a ums_context between several ums_schedulers

Macros

- #define **UMS_THREAD_STATE_IDLE** 0
- #define **UMS_THREAD_STATE_RUNNING** 1
- #define **UMS_THREAD_STATE_ENDED** 2
- #define INIT_UMS_CONTEXT(p_ums_context, p_routine, p_args)

    ums_context's constructor
- #define DESTROY_UMS_CONTEXT(p_ums_context)

    ums_context's destructor
- #define ums_context_register_as_thread(p_ums_context, p_task_struct, pid_sched)

    Register information about the actual thread used by the ums_context.
- #define ums_context_unregister_as_thread(p_ums_context)

    Unregister information about the actual thread used by the ums_context.
- #define ums_context_printable_state(p_ums_context) _ums_context_printable_state(p_ums_context)

    State of a ums_context as string.
- #define INIT_UMS_CONTEXT_SL(p_ums_context_sl, p_ums_context_in)

    ums_context_sl constructor
- #define DESTROY_UMS_CONTEXT_SL(p_ums_context_sl)

    ums_context_sl destructor
- #define ums_context_sl_set_assigned(p_ums_context_sl, assigned_in)

set "assigned" field of a ums_context_sl in a secure way
- #define ums_context_sl_get_assigned(p_ums_context_sl, p_assigned)

  read "assigned" field of a ums_context_sl in a secure way
- #define ums_context_sl_try_to_acquire(p_ums_context_sl, p_res)

  try to set "assigned" field of a ums_context_sl
- #define ums_context_update_run_time_start_slot(p_ums_context)

  update "ums_run_time" field of the ums_context To be called at the beginning of the slot
- #define ums_context_update_run_time_end_slot(p_ums_context)

  update "ums_run_time" field of the ums_context To be called at the end of the slot
- #define ums_context_get_run_time_ms(p_ums_context) jiffies_to_msecs((p_ums_context)->ums_run_time)

  get run time of the ums_context in milliseconds
- #define PRINTK_UMS_CONTEXT(p_obj, PREFIX)

  printK a ums_context
- #define PRINTK_UMS_CONTEXT_SL(p_obj, PREFIX)

  printK ums_context_sl

Typedefs

- typedef struct ums_context_t ums_context_t

  Represents a ums_context.
- typedef struct ums_context_sl_t ums_context_sl_t

  ums_context_SpinLock is used to protect a ums_context between several ums_schedulers

4.9.1 Detailed Description

This file contains definitions and functions of objects related to a ums_context

4.9.2 Macro Definition Documentation

4.9.2.1 DESTROY_UMS_CONTEXT

```
#define DESTROY_UMS_CONTEXT(
            p_ums_context )
```

Value:
```
do{ \
    (p_ums_context)->id = -1;     \
    (p_ums_context)->task_struct = NULL;     \
    (p_ums_context)->routine = NULL;    \
    (p_ums_context)->args = NULL; \
    (p_ums_context)->proc_entry = NULL; \
    (p_ums_context)->num_switch = 0; \
    (p_ums_context)->state = UMS_THREAD_STATE_IDLE; \
    (p_ums_context)->ums_run_time = 0; \
    (p_ums_context)->start_time_last_slot = 0; \
}while(0)
```

ums_context's destructor

Parameters

p_ums_context	pointer to a NON-NULL ums_context

Definition at line 82 of file ums_context.h.

4.9.2.2 DESTROY_UMS_CONTEXT_SL

```
#define DESTROY_UMS_CONTEXT_SL(
              p_ums_context_sl )
```

Value:

```
    do{ \
        (p_ums_context_sl)->id = -1; \
        \
        (p_ums_context_sl)->assigned = false;   \
        \
        (p_ums_context_sl)->ums_context = NULL;   \
    }while(0)
```

ums_context_sl destructor

Parameters

p_ums_context←_sl	NON-NULL ums_context_sl pointer

Definition at line 191 of file ums_context.h.

4.9.2.3 INIT_UMS_CONTEXT

```
#define INIT_UMS_CONTEXT(
              p_ums_context,
              p_routine,
              p_args )
```

Value:

```
    do{ \
        (p_ums_context)->id = -1;    \
        (p_ums_context)->task_struct = NULL;    \
        (p_ums_context)->routine = p_routine;   \
        (p_ums_context)->args = p_args; \
        (p_ums_context)->proc_entry = NULL; \
        (p_ums_context)->num_switch = 0; \
        (p_ums_context)->state = UMS_THREAD_STATE_IDLE; \
        (p_ums_context)->ums_run_time = 0; \
        (p_ums_context)->start_time_last_slot = 0; \
    }while(0)
```

ums_context's constructor

Parameters

p_ums_context	pointer to a NON-NULL ums_context
p_routine	pointer to user's routine (type: void∗ (*routine)(void* args))
p_args	args for user's routine

Definition at line 63 of file ums_context.h.

4.9.2.4 INIT_UMS_CONTEXT_SL

```
#define INIT_UMS_CONTEXT_SL(
                p_ums_context_sl,
                p_ums_context_in )
```

Value:

```
    do{ \
        (p_ums_context_sl)->id = -1; \
        \
        (p_ums_context_sl)->assigned = false;    \
        spin_lock_init(&(p_ums_context_sl)->assigned_spin_lock);  \
        \
        (p_ums_context_sl)->ums_context = p_ums_context_in;    \
    }while(0)
```

ums_context_sl constructor

Parameters

p_ums_context↩_sl	NON-NULL ums_context_sl pointer
p_ums_context↩_in	descriptor of the ums_context managed

Definition at line 176 of file ums_context.h.

4.9.2.5 PRINTK_UMS_CONTEXT

```
#define PRINTK_UMS_CONTEXT(
                p_obj,
                PREFIX )
```

Value:

```
    do{ \
        char* __buff = kmalloc(64, GFP_KERNEL);     \
        if(snprintf_ums_context(__buff, 64, p_obj) > 64-4) \
            printk(KERN_DEBUG "\nPRINTK_UMS_CONTEXT overflow!!!\n");     \
        else printk(PREFIX "ums_context_t = \n{\n%s}\n", __buff);  \
        kfree(__buff);  \
    }while(0)
```

printK a ums_context

Definition at line 316 of file ums_context.h.

4.9.2.6 PRINTK_UMS_CONTEXT_SL

```
#define PRINTK_UMS_CONTEXT_SL(
              p_obj,
              PREFIX )
```

Value:

```
    do{ \
        char* __buff = kmalloc(512, GFP_KERNEL);    \
        if( snprintf_ums_context_sl(__buff, 512, p_obj) > 512-4) \
            printk(KERN_DEBUG "\n PRINTK_UMS_CONTEXT_SL() overflow!!!\n");    \
        else  printk(PREFIX "ums_context_sl_t = \n{\n%s}\n", __buff);   \
        kfree(__buff);  \
    }while(0)
```

printK ums_context_sl

Definition at line 354 of file ums_context.h.

4.9.2.7 ums_context_get_run_time_ms

```
#define ums_context_get_run_time_ms(
              p_ums_context ) jiffies_to_msecs((p_ums_context)->ums_run_time)
```

get run time of the ums_context in milliseconds

Parameters

p_ums_context	NON-NULL pointer to ums_context

Returns

run time as unsigned int

Definition at line 286 of file ums_context.h.

4.9.2.8 ums_context_printable_state

```
#define ums_context_printable_state(
              p_ums_context ) _ums_context_printable_state(p_ums_context)
```

State of a ums_context as string.

Parameters

p_ums_context	Pointer to a NON-NULL ums_context

Returns

State as a human-readable string

Definition at line 148 of file ums_context.h.

4.9.2.9 ums_context_register_as_thread

```
#define ums_context_register_as_thread(
              p_ums_context,
              p_task_struct,
              pid_sched )
```

Value:

```
    do{ \
        (p_ums_context)->task_struct = p_task_struct;   \
        (p_ums_context)->pid = (p_task_struct)->pid;    \
        (p_ums_context)->pid_scheduler = pid_sched; \
    }while(0)
```

Register information about the actual thread used by the ums_context.

Parameters

p_ums_context	pointer to a NON-NULL ums_context
p_task_struct	pointer to thread's task_struct
pid_sched	pid of the scheduler that manages the ums_context

Definition at line 104 of file ums_context.h.

4.9.2.10 ums_context_sl_get_assigned

```
#define ums_context_sl_get_assigned(
              p_ums_context_sl,
              p_assigned )
```

Value:

```
    do{\
        spin_lock(&((p_ums_context_sl)->assigned_spin_lock)); \
        *(p_assigned) = (p_ums_context_sl)->assigned;    \
        spin_unlock(&((p_ums_context_sl)->assigned_spin_lock));   \
    }while(0)
```

read "assigned" field of a ums_context_sl in a secure way

Parameters

p_ums_context↩_sl	NON-NULL ums_context_sl
p_assigned	output, pointer to a bool

Definition at line 222 of file ums_context.h.

4.9.2.11 ums_context_sl_set_assigned

```
#define ums_context_sl_set_assigned(
              p_ums_context_sl,
              assigned_in )
```

Value:
```
    do{\
        spin_lock(&((p_ums_context_sl)->assigned_spin_lock)); \
        (p_ums_context_sl)->assigned = assigned_in;    \
        spin_unlock(&((p_ums_context_sl)->assigned_spin_lock));   \
    }while(0)
```

set "assigned" field of a ums_context_sl in a secure way

Parameters

p_ums_context_sl	NON-NULL ums_context_sl
assigned_in	boolean value to set

Definition at line 208 of file ums_context.h.

4.9.2.12 ums_context_sl_try_to_acquire

```
#define ums_context_sl_try_to_acquire(
              p_ums_context_sl,
              p_res )
```

Value:
```
    do{\
        spin_lock(&((p_ums_context_sl)->assigned_spin_lock)); \
        if(likely((p_ums_context_sl)->assigned == false)){  \
            (p_ums_context_sl)->assigned = true;    \
            *(p_res) = true;    \
        }   \
        else    \
            *(p_res) = false;   \
        spin_unlock(&((p_ums_context_sl)->assigned_spin_lock));   \
    }while(0)
```

try to set "assigned" field of a ums_context_sl

Parameters

p_ums_context_sl	NON-NULL ums_context_sl
p_res	output, true if the context has been acquired, false otherwise

Definition at line 237 of file ums_context.h.

4.9.2.13 ums_context_unregister_as_thread

```
#define ums_context_unregister_as_thread(
              p_ums_context )
```

Value:

```
do{ \
    (p_ums_context)->task_struct = NULL;    \
    (p_ums_context)->pid = 0;     \
    (p_ums_context)->pid_scheduler = 0; \
}while(0)
```

Unregister information about the actual thread used by the ums_context.

Parameters

p_ums_context	pointer to a NON-NULL ums_context

Definition at line 116 of file ums_context.h.

4.9.2.14 ums_context_update_run_time_end_slot

```
#define ums_context_update_run_time_end_slot(
              p_ums_context )
```

Value:

```
do{ \
    u64 time_now = get_jiffies_64();     \
    (p_ums_context)->ums_run_time += time_now-(p_ums_context)->start_time_last_slot; \
    (p_ums_context)->start_time_last_slot = 0; \
}while(0)
```

update "ums_run_time" field of the ums_context To be called at the end of the slot

Parameters

p_ums_context	NON-NULL pointer to ums_context

Definition at line 269 of file ums_context.h.

4.9.2.15 ums_context_update_run_time_start_slot

```
#define ums_context_update_run_time_start_slot(
              p_ums_context )
```

Value:

```
do{ \
    (p_ums_context)->start_time_last_slot = get_jiffies_64();     \
}while(0)
```

update "ums_run_time" field of the ums_context To be called at the beginning of the slot

Parameters

p_ums_context	NON-NULL pointer to ums_context

Definition at line 258 of file ums_context.h.

4.9.3 Typedef Documentation

4.9.3.1 ums_context_sl_t

```
typedef struct ums_context_sl_t ums_context_sl_t
```

ums_context_SpinLock is used to protect a ums_context between several ums_schedulers

4.9.3.2 ums_context_t

```
typedef struct ums_context_t ums_context_t
```

Represents a ums_context.

4.10 src/UMS/UMS_LKM/ums_hashtable.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/idr.h>
#include <linux/hashtable.h>
#include <linux/spinlock.h>
#include <stdbool.h>
#include <linux/list.h>
#include <linux/rwlock.h>
#include "../common/ums_types.h"
#include "ums_context.h"
#include "ums_completion_lsit.h"
#include "ums_scheduler.h"
#include <linux/proc_fs.h>
#include "ums_proc.h"
#include "ums_process.h"
```

Macros

- #define **UMS_HASHTABLE_HASH_BITS** 10 /** size of hashtable = 2^HASH_BITS */
- #define UMS_HASHTABLE_INIT() hash_init(ums_hashtable);
- #define ums_hashtable_add_process(p_ums_process)

  add a ums_process to the ums_hashtable
- #define ums_hashtable_remove_process(p_ums_process)

  remove a ums_process from the ums_hashtable
- #define ums_hashtable_get_process(tgid, p_ums_process)

  get a ums_process from its tgid
- #define ums_hashtable_create_process(tgid)

  create a new ums_process and add it to the ums_hashtable
- #define ums_hashtable_delete_process(tgid)

  delete a ums_process from the hashtable
- #define PRINTK_UMS_HASHTABLE(ignore)

  printK ums_hashtable

Functions

- **DECLARE_HASHTABLE** (ums_hashtable, UMS_HASHTABLE_HASH_BITS)
- int **test** (void)

Variables

- rwlock_t **ums_hashtable_rwlock**

4.10.1 Detailed Description

This file contains definitions and functions related to the ums_hashtable

4.10.2 Macro Definition Documentation

4.10.2.1 PRINTK_UMS_HASHTABLE

```
#define PRINTK_UMS_HASHTABLE(
                ignore )
```

Value:
```
    do{ \
        printk("################################### ums_hashtable ###################################\n"); \
        printk_ums_hashtable(); \
        printk("####################################################################################\n"); \
    }while(0)
```

printK ums_hashtable

Definition at line 176 of file ums_hashtable.h.

4.10.2.2 ums_hashtable_add_process

```
#define ums_hashtable_add_process(
              p_ums_process )
```

Value:
```
    write_lock(&ums_hashtable_rwlock);  \
        hash_add(ums_hashtable, &((p_ums_process)->hlist), (p_ums_process)->key);   \
    write_unlock(&ums_hashtable_rwlock);    \
```

add a ums_process to the ums_hashtable

Parameters

p_ums_process	NON-NULL pointer to a ums_process

NOTE: The key will be (p_ums_process)->key

Definition at line 58 of file ums_hashtable.h.

4.10.2.3 ums_hashtable_create_process

```
#define ums_hashtable_create_process(
              tgid )
```

Value:
```
    do{ \
        ums_process_t* item = kmalloc(sizeof(ums_process_t), GFP_KERNEL);   \
        INIT_UMS_PROCESS(item, tgid);   \
        \
        write_lock(&ums_hashtable_rwlock);  \
            hash_add(ums_hashtable, &item->hlist, item->key);   \
        write_unlock(&ums_hashtable_rwlock);    \
        \
        ums_proc_add_process(item->proc_entry, item->proc_entry_main_scheds, tgid); \
    }while(0)
```

create a new ums_process and add it to the ums_hashtable

Parameters

tgid	ums_process's tgid, namely the tgid of the actual Linux process

Definition at line 104 of file ums_hashtable.h.

4.10.2.4 ums_hashtable_delete_process

```
#define ums_hashtable_delete_process(
              tgid )
```

Value:

```
    do{ \
        ums_process_t* ums_process; \
        ums_hashtable_get_process(tgid, ums_process);   \
        if(likely(ums_process != NULL)){    \
            ums_proc_remove_process(ums_process->proc_entry, ums_process->proc_entry_main_scheds);    \
            \
            write_lock(&ums_hashtable_rwlock);  \
                hash_del(&ums_process->hlist);  \
            write_unlock(&ums_hashtable_rwlock);    \
            \
            DESTROY_UMS_PROCESS(ums_process);   \
            kfree(ums_process); \
        }   \
    }while(0)
```

delete a ums_process from the hashtable

Parameters

tgid	tgid of the ums_process to delete

NOTE: This function must be used only if the ums_process has been created by ums_hashtable_create_process()

Definition at line 123 of file ums_hashtable.h.

4.10.2.5 ums_hashtable_get_process

```
#define ums_hashtable_get_process(
                tgid,
                p_ums_process )
```

Value:

```
    do{ \
    ums_process_t* current_ums_process = NULL;  \
    read_lock(&ums_hashtable_rwlock);   \
        /*iterate a bucket*/    \
        hash_for_each_possible(ums_hashtable, current_ums_process, hlist, tgid){    \
            if(likely(current_ums_process && current_ums_process->key == tgid))    break;\
        }   \
    read_unlock(&ums_hashtable_rwlock); \
    \
    p_ums_process = (likely(current_ums_process && current_ums_process->key == tgid)) ? current_ums_process
       : NULL; \
    }while(0)
```

get a ums_process from its tgid

Parameters

tgid	ums_process's tgid (key in the hashtable)
p_ums_process	output, pointer to a ums_process

Definition at line 82 of file ums_hashtable.h.

4.10.2.6 UMS_HASHTABLE_INIT

```
#define UMS_HASHTABLE_INIT( ) hash_init(ums_hashtable);
```

init ums_hashtable

Definition at line 48 of file ums_hashtable.h.

4.10.2.7 ums_hashtable_remove_process

```
#define ums_hashtable_remove_process(
              p_ums_process )
```

Value:
```
    write_lock(&ums_hashtable_rwlock);  \
        hash_del(&((p_ums_process)->hlist));   \
    write_unlock(&ums_hashtable_rwlock);    \
```

remove a ums_process from the ums_hashtable

Parameters

p_ums_process	NON-NULL pointer to a ums_process

Definition at line 68 of file ums_hashtable.h.

4.11 src/UMS/UMS_LKM/ums_proc.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/proc_fs.h>
#include <asm/uaccess.h>
#include "ums_hashtable.h"
#include "ums_process.h"
#include "ums_context.h"
#include "ums_completion_lsit.h"
```

Macros

- #define __sched_file_to_sched_pid(p_file, p_pid_out)
  get pid of the scheduler to which "file" refers to, in /proc/ums
- #define __sched_file_to_tgid(p_file, p_tgid_out)
  get tgid of the process to which "file" refers to, in /proc/ums
- #define **__INFO_FILE_BUFF_SIZE** 128
- #define __worker_file_to_ucd(p_file, p_ucd)
  get ums_context_descriptor of the ums_context to which "p_file" refers to, in /proc/ums
- #define __worker_file_to_sched_pid(p_file, p_pid_out)
  get pid of the scheduler to which "file" refers to, in /proc/ums
- #define __worker_file_to_tgid(p_file, p_tgid_out)
  get tgid of the process to which "file" refers to, in /proc/ums

- #define **__INFO_WORKER_BUFF_SIZE** 128
- #define ums_proc_mount()

    make /proc/ums directory
- #define ums_proc_unmount()

    remove /proc/ums directory
- #define ums_proc_add_process(p_pd_proc_out, p_pd_scheds_out, pid)

    add a process in /proc/ums
- #define ums_proc_remove_process(p_pd_proc_in, p_pd_sched_in)

    remove a process in /proc/ums
- #define ums_proc_add_scheduler(p_pd_scheds_main_in, sched_id, p_pd_sched_out, p_info_out, p_pd_↩ workers_main_out)

    add a scheduler in /proc/ums/<tgid>
- #define ums_proc_remove_scheduler(p_pd_sched_in, p_info_in, p_pd_workers_main_in)

    remove a scheduler in /proc/ums/<tgid>
- #define ums_proc_add_thread(p_pd_workers_main_in, id_thread_in, p_pd_thread_out)

    add a ums_context in /proc/ums/tgid/schedulers/<pid>/workers
- #define ums_proc_remove_thread(p_pd_thread_in)

    remove a ums_context from /proc/ums/tgid/schedulers/<pid>/workers

Functions

- ssize_t ums_scheduler_snprintf_info (pid_t tgid, pid_t sched_pid, char ∗buff, size_t buff_size)

    snprintf used to print info about scheduler in /proc
- ssize_t ums_scheduler_snprintf_worker (pid_t tgid, pid_t sched_pid, ums_context_descriptor_t ucd, char ∗buff, size_t buff_size)

    snprintf used to print info about a worker thread in /proc

Variables

- struct proc_dir_entry ∗ **ums_proc_ums_folder**

4.11.1 Detailed Description

This file contains functions used to manage /proc filesystem

4.11.2 Macro Definition Documentation

4.11.2.1 __sched_file_to_sched_pid

```
#define __sched_file_to_sched_pid(
                p_file,
                p_pid_out )
```

Value:

```
    do{ \
        long tmp;   \
        int res = kstrtol((p_file)->f_path.dentry->d_parent->d_iname, 10, &tmp); \
        if(likely(res==0))    \
            *(p_pid_out) = (pid_t)tmp;   \
        else    \
            *(p_pid_out) = -1;   \
    }while(0)
```

get pid of the scheduler to which "file" refers to, in /proc/ums

Parameters

p_file	pointer to a struct file
p_pid_out	output, pointer to a pid_t

Definition at line 55 of file ums_proc.h.

4.11.2.2 __sched_file_to_tgid

```
#define __sched_file_to_tgid(
              p_file,
              p_tgid_out )
```

Value:

```
    do{ \
        long tmp;   \
        int res = kstrtol((p_file)->f_path.dentry->d_parent->d_parent->d_parent->d_iname, 10, &tmp);    \
        if(likely(res==0))  \
            *(p_tgid_out) = (pid_t)tmp; \
        else \
            *(p_tgid_out) = -1; \
    }while(0)
```

get tgid of the process to which "file" refers to, in /proc/ums

Definition at line 69 of file ums_proc.h.

4.11.2.3 __worker_file_to_sched_pid

```
#define __worker_file_to_sched_pid(
              p_file,
              p_pid_out )
```

Value:

```
    do{ \
        long tmp;   \
        int res = kstrtol((p_file)->f_path.dentry->d_parent->d_parent->d_iname, 10, &tmp); \
        if(likely(res==0))   \
            *(p_pid_out) = (pid_t)tmp;  \
        else    \
            *(p_pid_out) = -1;  \
    }while(0)
```

get pid of the scheduler to which "file" refers to, in /proc/ums

Parameters

p_file	pointer to a struct file
p_pid_out	output, pointer to a pid_t

Definition at line 136 of file ums_proc.h.

4.11.2.4 __worker_file_to_tgid

```
#define __worker_file_to_tgid(
                p_file,
                p_tgid_out )
```

Value:
```
    do{ \
        long tmp;   \
        int res = kstrtol((p_file)->f_path.dentry->d_parent->d_parent->d_parent->d_parent->d_iname, 10,
       &tmp);    \
        if(likely(res==0))  \
            *(p_tgid_out) = (pid_t)tmp; \
        else \
            *(p_tgid_out) = -1; \
    }while(0)
```

get tgid of the process to which "file" refers to, in /proc/ums

Parameters

p_file	pointer to a struct file
p_tgid_out	output, pointer to a pid_t

Definition at line 152 of file ums_proc.h.

4.11.2.5 __worker_file_to_ucd

```
#define __worker_file_to_ucd(
                p_file,
                p_ucd )
```

Value:
```
    do{ \
        long tmp;   \
        int res = kstrtol((p_file)->f_path.dentry->d_iname, 10, &tmp); \
        if(likely(res==0))   \
            *(p_ucd) = (int)tmp;  \
        else    \
            *(p_ucd) = -1;  \
    }while(0)
```

get ums_context_descriptor of the ums_context to which "p_file" refers to, in /proc/ums

Parameters

p_file	pointer to a struct file
p_ucd	output, pointer to a ums_context_descriptor

Definition at line 120 of file ums_proc.h.

4.11.2.6 ums_proc_add_process

```
#define ums_proc_add_process(
              p_pd_proc_out,
              p_pd_scheds_out,
              pid )
```

Value:

```
    do{ \
        char __buff[32];    \
        sprintf(__buff, "%d", pid); \
        p_pd_proc_out = proc_mkdir(__buff, ums_proc_ums_folder);  \
        p_pd_scheds_out = proc_mkdir("schedulers", p_pd_proc_out);  \
    }while(0)
```

add a process in /proc/ums

Parameters

p_pd_proc_out	output, pointer proc_dir_entry of the new directory /proc/ums/<tgid>
p_pd_scheds_out	output, pointer proc_dir_entry of the new directory /proc/ums/<tgid>/schedulers
pid	pid of the process (tgid)

Definition at line 233 of file ums_proc.h.

4.11.2.7 ums_proc_add_scheduler

```
#define ums_proc_add_scheduler(
              p_pd_scheds_main_in,
              sched_id,
              p_pd_sched_out,
              p_info_out,
              p_pd_workers_main_out )
```

Value:

```
    do{ \
        char __buff[32];    \
        sprintf(__buff, "%d", sched_id);    \
        \
        p_pd_sched_out = proc_mkdir(__buff, p_pd_scheds_main_in);   \
        p_info_out = proc_create("info", S_IALLUGO, p_pd_sched_out, &sched_info_ops);    \
        p_pd_workers_main_out = proc_mkdir("workers", p_pd_sched_out);  \
    }while(0)
```

add a scheduler in /proc/ums/<tgid>

Parameters

p_pd_scheds_main_in	pointer proc_dir_entry of the directory /proc/ums/<tgid>/schedulers
sched_id	pid of the scheduler
p_pd_sched_out	output, pointer proc_dir_entry of the directory /proc/ums/<tgid>/schedulers/<sched_pid>
p_info_out	output, pointer proc_dir_entry of the file "info" in /proc/ums/<tgid>/schedulers/<sched_pid>
p_pd_workers_main_out	output, pointer proc_dir_entry of the folder /proc/ums/<tgid>/schedulers/<sched_pid>/workers

Definition at line 263 of file ums_proc.h.

4.11.2.8 ums_proc_add_thread

```
#define ums_proc_add_thread(
              p_pd_workers_main_in,
              id_thread_in,
              p_pd_thread_out )
```

Value:

```
    do{ \
        char __buff[32];    \
        sprintf(__buff, "%d", id_thread_in);    \
        p_pd_thread_out = proc_create(__buff, S_IALLUGO, p_pd_workers_main_in, &thread_ops);    \
    }while(0)
```

add a ums_context in /proc/ums/tgid/schedulers/<pid>/workers

Parameters

p_pd_workers_main↵_in	pointer proc_dir_entry of the folder /proc/ums/<tgid>/schedulers/<sched_pid>/workers
id_thread_in	ums_context_descriptor
p_pd_thread_out	output, proc_dir_entry of the file associated to ums_context in /proc/ums/tgid/schedulers/<pid>/workers

Definition at line 297 of file ums_proc.h.

4.11.2.9 ums_proc_mount

```
#define ums_proc_mount( )
```

Value:

```
    do{ \
        ums_proc_ums_folder = proc_mkdir("ums", NULL);  \
    }while(0)
```

make /proc/ums directory

Definition at line 212 of file ums_proc.h.

4.11.2.10 ums_proc_remove_process

```
#define ums_proc_remove_process(
              p_pd_proc_in,
              p_pd_sched_in )
```

Value:

```
    do{ \
        proc_remove(p_pd_sched_in); \
        proc_remove(p_pd_proc_in);  \
    }while(0)
```

remove a process in /proc/ums

Parameters

p_pd_proc_in	pointer proc_dir_entry of the directory /proc/ums/<tgid> to remove
p_pd_scheds↩ _in	pointer proc_dir_entry of the directory /proc/ums/<tgid>/schedulers to remove

Definition at line 247 of file ums_proc.h.

4.11.2.11 ums_proc_remove_scheduler

```
#define ums_proc_remove_scheduler(
              p_pd_sched_in,
              p_info_in,
              p_pd_workers_main_in )
```

Value:
```
    do{ \
        proc_remove(p_pd_workers_main_in);  \
        proc_remove(p_info_in);     \
        proc_remove(p_pd_sched_in); \
    }while(0)
```

remove a scheduler in /proc/ums/<tgid>

Parameters

p_pd_sched_in	pointer proc_dir_entry of the directory /proc/ums/<tgid>/schedulers/<sched_pid>
p_info_in	pointer proc_dir_entry of the file "info" in /proc/ums/<tgid>/schedulers/<sched_pid>
p_pd_workers_main↩ _in	pointer proc_dir_entry of the folder /proc/ums/<tgid>/schedulers/<sched_pid>/workers

Definition at line 281 of file ums_proc.h.

4.11.2.12 ums_proc_remove_thread

```
#define ums_proc_remove_thread(
              p_pd_thread_in )
```

Value:
```
    do{ \
        proc_remove(p_pd_thread_in);    \
    }while(0)
```

remove a ums_context from /proc/ums/tgid/schedulers/<pid>/workers

Parameters

p_pd_thread↩ _in	proc_dir_entry of the file associated to ums_context in /proc/ums/tgid/schedulers/<pid>/workers

Definition at line 310 of file ums_proc.h.

4.11.2.13 ums_proc_unmount

```
#define ums_proc_unmount( )
```

Value:
```
    do{ \
        proc_remove(ums_proc_ums_folder);   \
    }while(0)
```

remove /proc/ums directory

Definition at line 221 of file ums_proc.h.

4.11.3 Function Documentation

4.11.3.1 ums_scheduler_snprintf_info()

```
ssize_t ums_scheduler_snprintf_info (
             pid_t tgid,
             pid_t sched_pid,
             char * buff,
             size_t buff_size )
```

snprintf used to print info about scheduler in /proc

entry in /proc of "ums" folder

Parameters

tgid	tgid of the process
sched_pid	pid of the scheduler

Definition at line 8 of file ums_proc.c.

4.11.3.2 ums_scheduler_snprintf_worker()

```
ssize_t ums_scheduler_snprintf_worker (
             pid_t tgid,
             pid_t sched_pid,
             ums_context_descriptor_t ucd,
             char * buff,
             size_t buff_size )
```

snprintf used to print info about a worker thread in /proc

Parameters

tgid	tgid of the process
sched_pid	pid of the scheduler that manages the ums_context
ucd	descriptor of the ums_context

Definition at line 61 of file ums_proc.c.

4.12 src/UMS/UMS_LKM/ums_process.h File Reference

```
#include <linux/proc_fs.h>
#include "ums_scheduler.h"
```

Data Structures

- struct ums_process_t

  Represent a ums_process object.

Macros

- #define **UMS_PROCESS_COMPLETION_LIST_MIN_ID** 0 /∗∗ Lower value for a ums_completion_list descriptor ∗/
- #define **UMS_PROCESS_COMPLETION_LIST_MAX_ID** 127 /∗∗ Higher value for a ums_completion_list descriptor ∗/
- #define **UMS_PROCESS_UMS_CONTEXT_MIN_ID** 0 /∗∗ Lower value for a ums_context descriptor ∗/
- #define **UMS_PROCESS_UMS_CONTEXT_MAX_ID** 127 /∗∗ Higher value for a ums_context descriptor ∗/
- #define **HASHTABLE_UMS_SCHEDULERS_HASH_BITS** 6 /∗∗ size of hashtable = 2^HASH_BITS ∗/
- #define **HASHTABLE_UMS_THREADS_HASH_BITS** 6 /∗∗ size of hashtable = 2^HASH_BITS ∗/
- #define INIT_UMS_PROCESS(p_ums_process, key_in)

  ums_process constructor
- #define DESTROY_UMS_PROCESS(p_ums_process)

  ums_process destructor
- #define ums_process_add_scheduler_sl(p_ums_process, p_ums_scheduler_sl)

  add a ums_scheduler_sl object to the process
- #define ums_process_remove_scheduler_sl(p_ums_process, p_ums_scheduler_sl)

  remove a ums_scheduler_sl from the process
- #define ums_process_register_ums_thread(p_ums_process, p_ums_context)

  register a ums_context in hashtable_ums_thread of the process
- #define ums_process_unregister_ums_thread(p_ums_process, p_ums_context)

  unregister a ums_context from hashtable_ums_thread of the process
- #define ums_process_get_ums_thread(p_ums_process, key_in, p_ums_context_OUT)

  get a ums_context from hashtable_threads of the process by its pid
- #define ums_process_get_scheduler_sl(p_ums_process, key_in, p_ums_scheduler_sl_OUT)

  get a ums_scheduler_sl from the ums_hashtable
- #define ums_process_add_ums_context_sl(p_ums_process, p_ums_context_sl)

  add a ums_context_sl to idr_ums_context of the process
- #define ums_process_remove_ums_context_sl(p_ums_process, p_ums_context_sl)

remove a ums_context_sl from idr_ums_context of the ums_process

- #define ums_process_add_ums_completion_list_sl(p_ums_process, p_ums_completion_list_sl)

    add a ums_completion_list_sl to the process
- #define ums_process_remove_ums_completion_list_sl(p_ums_process, p_ums_completion_list_sl)

    remove a ums_completion_list_sl to the process
- #define ums_process_get_ums_completion_list_sl(p_ums_process, id, p_ums_completion_list_sl_OUT)

    get a ums_completion_list_sl object from the process
- #define ums_process_get_ums_context_sl(p_ums_process, id, p_ums_context_sl_OUT)

    get a ums_context_sl object from the process
- #define PRINTK_UMS_PROCESS(p_ums_process, ignore)

    printk a ums_process

Typedefs

- typedef struct ums_process_t ums_process_t

    Represent a ums_process object.

4.12.1 Detailed Description

This file contains definitions and functions of objects related to a ums_process

4.12.2 Macro Definition Documentation

4.12.2.1 DESTROY_UMS_PROCESS

```
#define DESTROY_UMS_PROCESS(
              p_ums_process )
```

Value:

```
    do {    \
        idr_destroy(&(p_ums_process)->idr_completion_list);       \
        \
        idr_destroy(&(p_ums_process)->idr_ums_context);       \
        \
        (p_ums_process)->key = 0;       \
        (p_ums_process)->proc_entry = NULL;   \
    }while(0)
```

ums_process destructor

Parameters

p_ums_process	NON-NULL pointer to a ums_process

Definition at line 74 of file ums_process.h.

4.12.2.2 INIT_UMS_PROCESS

```
#define INIT_UMS_PROCESS(
                p_ums_process,
                key_in )
```

Value:
```
    do {                            \
        (p_ums_process)->key = key_in;       \
        (p_ums_process)->proc_entry = NULL;  \
        \
        hash_init((p_ums_process)->hashtable_ums_schedulers);   \
        rwlock_init(&(p_ums_process)->hashtable_ums_schedulers_rwlock);    \
        \
        hash_init((p_ums_process)->hashtable_ums_threads);  \
        rwlock_init(&(p_ums_process)->hashtable_ums_threads_rwlock);   \
        \
        idr_init(&(p_ums_process)->idr_completion_list);       \
        rwlock_init(&(p_ums_process)->idr_completion_list_rwlock); \
        \
        idr_init(&(p_ums_process)->idr_ums_context);       \
        rwlock_init(&(p_ums_process)->idr_ums_context_rwlock); \
    } while (0)
```

ums_process constructor

Parameters

p_ums_process	NON-NULL pointer to a ums_process
key_in	key used in the ums_hashtable, it corresponds to <tgid>

Definition at line 51 of file ums_process.h.

4.12.2.3 PRINTK_UMS_PROCESS

```
#define PRINTK_UMS_PROCESS(
                p_ums_process,
                ignore )
```

Value:
```
    do{\
        printk_ums_process(p_ums_process);    \
    }while(0)
```

printk a ums_process

Definition at line 392 of file ums_process.h.

4.12.2.4 ums_process_add_scheduler_sl

```
#define ums_process_add_scheduler_sl(
                p_ums_process,
                p_ums_scheduler_sl )
```

Value:

```
    do{ \
        write_lock(&((p_ums_process)->hashtable_ums_schedulers_rwlock));    \
            hash_add(p_ums_process->hashtable_ums_schedulers, &((p_ums_scheduler_sl)->hlist),
       (p_ums_scheduler_sl)->key);   \
        write_unlock(&((p_ums_process)->hashtable_ums_schedulers_rwlock));  \
        \
        ums_proc_add_scheduler((p_ums_process)->proc_entry_main_scheds, (p_ums_scheduler_sl)->key,
       (p_ums_scheduler_sl)->proc_entry, (p_ums_scheduler_sl)->proc_entry_info,
       (p_ums_scheduler_sl)->proc_entry_main_workers);    \
        \
    }while(0)
```

add a ums_scheduler_sl object to the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_scheduler↩ _sl	NON-NULL pointer to the ums_scheduler_sl to add

Definition at line 92 of file ums_process.h.

4.12.2.5 ums_process_add_ums_completion_list_sl

```
#define ums_process_add_ums_completion_list_sl(
                p_ums_process,
                p_ums_completion_list_sl )
```

Value:

```
    do{\
        write_lock(&((p_ums_process)->idr_completion_list_rwlock));     \
            (p_ums_completion_list_sl)->id = idr_alloc(&((p_ums_process)->idr_completion_list),
       p_ums_completion_list_sl, UMS_PROCESS_COMPLETION_LIST_MIN_ID, UMS_PROCESS_COMPLETION_LIST_MAX_ID,
       GFP_KERNEL); \
        write_unlock(&((p_ums_process)->idr_completion_list_rwlock));   \
    }while(0)
```

add a ums_completion_list_sl to the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_completion_list↩ _sl	NON-NULL pointer to the object to add

Definition at line 235 of file ums_process.h.

4.12.2.6 ums_process_add_ums_context_sl

```
#define ums_process_add_ums_context_sl(
                p_ums_process,
                p_ums_context_sl )
```

Value:
```
    do{\
        write_lock(&((p_ums_process)->idr_ums_context_rwlock));      \
            (p_ums_context_sl)->id = idr_alloc(&((p_ums_process)->idr_ums_context), p_ums_context_sl,
       UMS_PROCESS_UMS_CONTEXT_MIN_ID, UMS_PROCESS_UMS_CONTEXT_MAX_ID, GFP_KERNEL); \
            (p_ums_context_sl)->ums_context->id = (p_ums_context_sl)->id; \
        write_unlock(&((p_ums_process)->idr_ums_context_rwlock));    \
    }while(0)
```

add a ums_context_sl to idr_ums_context of the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_context↵_sl	NON-NULL pointer ums_context_sl to add

Definition at line 203 of file ums_process.h.

4.12.2.7 ums_process_get_scheduler_sl

```
#define ums_process_get_scheduler_sl(
                p_ums_process,
                key_in,
                p_ums_scheduler_sl_OUT )
```

Value:
```
    do{ \
    ums_scheduler_sl_t* current_ums_scheduler_sl = NULL;  \
    read_lock(&((p_ums_process)->hashtable_ums_schedulers_rwlock));   \
        /*iterate a bucket*/    \
        hash_for_each_possible((p_ums_process)->hashtable_ums_schedulers, current_ums_scheduler_sl, hlist,
       key_in){    \
            if(likely(current_ums_scheduler_sl && current_ums_scheduler_sl->key == key_in))    break;\
        }   \
    read_unlock(&((p_ums_process)->hashtable_ums_schedulers_rwlock)); \
    \
    p_ums_scheduler_sl_OUT = (likely(current_ums_scheduler_sl && current_ums_scheduler_sl->key == key_in)) ?
       current_ums_scheduler_sl : NULL; \
    }while(0)
```

get a ums_scheduler_sl from the ums_hashtable

Parameters

p_ums_process	NON-NULL pointer to a ums_process
key_in	pid of the scheduler
p_ums_scheduler_sl_OUT	output, pointer to a ums_scheduler_sl

Definition at line 181 of file ums_process.h.

4.12.2.8 ums_process_get_ums_completion_list_sl

```
#define ums_process_get_ums_completion_list_sl(
                p_ums_process,
                id,
                p_ums_completion_list_sl_OUT )
```

Value:

```
    do{ \
        read_lock(&((p_ums_process)->idr_completion_list_rwlock));      \
            p_ums_completion_list_sl_OUT = idr_find(&((p_ums_process)->idr_completion_list), id); \
        read_unlock(&((p_ums_process)->idr_completion_list_rwlock));    \
    }while(0)
```

get a ums_completion_list_sl object from the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
id	ums_completion_list_sl descriptor
p_ums_completion_list_sl_OUT	output, pointer object to get

Definition at line 265 of file ums_process.h.

4.12.2.9 ums_process_get_ums_context_sl

```
#define ums_process_get_ums_context_sl(
                p_ums_process,
                id,
                p_ums_context_sl_OUT )
```

Value:

```
    do{ \
        read_lock(&((p_ums_process)->idr_ums_context_rwlock));      \
            p_ums_context_sl_OUT = idr_find(&((p_ums_process)->idr_ums_context), id); \
        read_unlock(&((p_ums_process)->idr_ums_context_rwlock));    \
    }while(0)
```

get a ums_context_sl object from the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
id	ums_context descriptor
p_ums_context_sl_OUT	output, pointer object to get

Definition at line 281 of file ums_process.h.

4.12.2.10 ums_process_get_ums_thread

```
#define ums_process_get_ums_thread(
              p_ums_process,
              key_in,
              p_ums_context_OUT )
```

Value:

```
    do{ \
    ums_context_t* current_ums_context = NULL;  \
    read_lock(&((p_ums_process)->hashtable_ums_threads_rwlock));   \
        /*iterate a bucket*/    \
        hash_for_each_possible((p_ums_process)->hashtable_ums_threads, current_ums_context, hlist, key_in){
         \
            if(likely(current_ums_context && current_ums_context->pid == key_in))    break;\
        }   \
    read_unlock(&((p_ums_process)->hashtable_ums_threads_rwlock)); \
    \
    p_ums_context_OUT = (likely(current_ums_context && current_ums_context->pid == key_in)) ?
       current_ums_context : NULL; \
    }while(0)
```

get a ums_context from hashtable_threads of the process by its pid

Parameters

p_ums_process	NON-NULL pointer to a ums_process
key_in	pid of the thread
p_ums_context_OUT	output, pointer to a ums_context

Definition at line 157 of file ums_process.h.

4.12.2.11 ums_process_register_ums_thread

```
#define ums_process_register_ums_thread(
              p_ums_process,
              p_ums_context )
```

Value:

```
    do{ \
        write_lock(&((p_ums_process)->hashtable_ums_threads_rwlock));    \
            hash_add(p_ums_process->hashtable_ums_threads, &((p_ums_context)->hlist), (p_ums_context)->pid);
         \
        write_unlock(&((p_ums_process)->hashtable_ums_threads_rwlock));  \
    }while(0)
```

register a ums_context in hashtable_ums_thread of the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_context	NON-NULL pointer to a ums_context to add

Definition at line 127 of file ums_process.h.

4.12.2.12 ums_process_remove_scheduler_sl

```
#define ums_process_remove_scheduler_sl(
            p_ums_process,
            p_ums_scheduler_sl )
```

Value:

```
do{ \
    write_lock(&((p_ums_process)->hashtable_ums_schedulers_rwlock));    \
        hash_del(&((p_ums_scheduler_sl)->hlist));   \
    write_unlock(&((p_ums_process)->hashtable_ums_schedulers_rwlock));  \
    \
    ums_proc_remove_scheduler((p_ums_scheduler_sl)->proc_entry, (p_ums_scheduler_sl)->proc_entry_info,
   (p_ums_scheduler_sl)->proc_entry_main_workers);    \
    \
}while(0)
```

remove a ums_scheduler_sl from the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_scheduler←_sl	NON-NULL pointer to the ums_scheduler to remove

Definition at line 108 of file ums_process.h.

4.12.2.13 ums_process_remove_ums_completion_list_sl

```
#define ums_process_remove_ums_completion_list_sl(
            p_ums_process,
            p_ums_completion_list_sl )
```

Value:

```
do{\
    write_lock(&((p_ums_process)->idr_completion_list_rwlock));     \
        idr_remove(&((p_ums_process)->idr_completion_list), (p_ums_completion_list_sl)->id); \
        (p_ums_completion_list_sl)->id = -1;    \
    write_unlock(&((p_ums_process)->idr_completion_list_rwlock));   \
}while(0)
```

remove a ums_completion_list_sl to the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_completion_list←_sl	NON-NULL pointer to the object to remove

Definition at line 248 of file ums_process.h.

4.12.2.14 ums_process_remove_ums_context_sl

```
#define ums_process_remove_ums_context_sl(
                p_ums_process,
                p_ums_context_sl )
```

Value:
```
    do{ \
        write_lock(&((p_ums_process)->idr_ums_context_rwlock));    \
            idr_remove(&((p_ums_process)->idr_ums_context), (p_ums_context_sl)->id); \
            (p_ums_context_sl)->id = -1;     \
            (p_ums_context_sl)->ums_context->id = -1; \
        write_unlock(&((p_ums_process)->idr_ums_context_rwlock));    \
    }while(0)
```

remove a ums_context_sl from idr_ums_context of the ums_process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_context←_sl	NON-NULL pointer to the ums_context_sl to remove

Definition at line 218 of file ums_process.h.

4.12.2.15 ums_process_unregister_ums_thread

```
#define ums_process_unregister_ums_thread(
                p_ums_process,
                p_ums_context )
```

Value:
```
    do{ \
        write_lock(&((p_ums_process)->hashtable_ums_threads_rwlock));     \
            hash_del(&((p_ums_context)->hlist));    \
        write_unlock(&((p_ums_process)->hashtable_ums_threads_rwlock));   \
    }while(0)
```

unregister a ums_context from hashtable_ums_thread of the process

Parameters

p_ums_process	NON-NULL pointer to a ums_process
p_ums_context	NON-NULL pointer to the ums_context to remove

Definition at line 141 of file ums_process.h.

4.12.3 Typedef Documentation

4.12.3.1 ums_process_t

```
typedef struct ums_process_t ums_process_t
```

Represent a ums_process object.

4.13 src/UMS/UMS_LKM/ums_scheduler.h File Reference

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/idr.h>
#include <linux/hashtable.h>
#include <linux/spinlock.h>
#include <stdbool.h>
#include <linux/list.h>
#include <linux/rwlock.h>
#include "../common/ums_types.h"
#include "ums_context.h"
#include "ums_completion_lsit.h"
#include <linux/proc_fs.h>
```

Data Structures

- struct ums_scheduler_t

    object that represent a ums_scheduler
- struct ums_scheduler_sl_t

    object used to arrange a ums_scheduler in a hashtable and to protect it with a spin_lock
- struct idr_for_each_handler_arg_t

Macros

- #define INIT_UMS_SCHEDULER(p_ums_scheduler, p_scheduler_task_struct_in, p_completion_list_in)

    ums_scheduler constructor
- #define DESTROY_UMS_SCHEDULER(p_ums_scheduler)

    ums_scheduler deconstructor
- #define ums_scheduler_ready_list_add(p_ums_scheduler, p_ums_context)

    add a ums_context to ready list of the scheduler
- #define ums_scheduler_ready_list_remove(p_ums_scheduler, p_ums_context)

    remove a ums_context from the ready list of the scheduler
- #define ums_scheduler_ready_list_remove_first(p_ums_scheduler, p_ums_context_OUT)

    remove first ums_context from the ready list
- #define ums_scheduler_completion_list_start_iteration(p_ums_scheduler, p_ums_completion_list_item_out)

    start to iterate the completion_list
- #define ums_scheduler_completion_list_iteration_get_current(p_ums_scheduler, p_ums_completion_list_↩
item_out)

    get current element during the iteration of the completion_list
- #define ums_scheduler_completion_list_iterate(p_ums_scheduler, p_ums_completion_list_item_out)

    get next ums_completion_list_item during navigation
- #define ums_scheduler_completion_list_iterate_end(p_ums_scheduler)

    end to iterate the completion_list
- #define ums_scheduler_ready_list_start_iteration(p_ums_scheduler, p_ums_context_out)

    start to iterate the ready_list
- #define ums_scheduler_ready_list_iterate(p_ums_scheduler, p_ums_context_out)

    get next ums_context during navigation

- #define ums_scheduler_ready_list_iterate_end(p_ums_scheduler)

  end to iterate the ready_list
- #define ums_scheduler_set_reason_end_sheduler(p_ums_scheduler)

  set reason of the next scheduler call
- #define INIT_UMS_SCHEDULER_SL(p_ums_scheduler_sl, key_in, p_ums_scheduler_in)

  ums_scheduler_sl constructor
- #define DESTROY_UMS_SCHEDULER_SL(p_ums_scheduler_sl)

  ums_scheduler_sl destructor
- #define ums_scheduler_sl_remove_scheduler(p_ums_scheduler_sl, p_ums_scheduler_OUT)

  remove ums_scheduler from the ums_scheduler_sl
- #define ums_scheduler_sl_lock_get_scheduler(p_ums_scheduler_sl, p_ums_scheduler_OUT)

  lock the ums_scheduler in the ums_scheduler_sl object
- #define ums_scheduler_sl_unlock_scheduler(p_ums_scheduler_sl)

  unlock the ums_scheduler in the ums_scheduler_sl object
- #define ums_scheduler_list_empty(p_list_head) list_empty(p_list_head)

  macro used to check if a list is empty
- #define PRINTK_UMS_SCHEDULER(p_obj, PREFIX)

  printK a ums_scheduler
- #define PRINTK_UMS_SCHEDULER_SL(p_obj, PREFIX)

  printK ums_scheduler_sl

Typedefs

- typedef struct ums_scheduler_t ums_scheduler_t

  object that represent a ums_scheduler
- typedef struct ums_scheduler_sl_t ums_scheduler_sl_t

  object used to arrange a ums_scheduler in a hashtable and to protect it with a spin_lock
- typedef struct idr_for_each_handler_arg_t **idr_for_each_handler_arg_t**

4.13.1 Detailed Description

This file contains definitions and functions of objects related to a ums_scheduler

4.13.2 Macro Definition Documentation

4.13.2.1 DESTROY_UMS_SCHEDULER

```
#define DESTROY_UMS_SCHEDULER(
                p_ums_scheduler )
```

Value:
```
    do{ \
        (p_ums_scheduler)->scheduler_task_struct = NULL;   \
        \
        (p_ums_scheduler)->completion_list = NULL;   \
        (p_ums_scheduler)->current_completion_list_item = NULL; \
        \
        (p_ums_scheduler)->current_ready_list_item = NULL;   \
        \
        (p_ums_scheduler)->running_thread = NULL;    \
        (p_ums_scheduler)->num_switch = 0;    \
        (p_ums_scheduler)->cpu_core = -1;    \
    }while(0)
```

ums_scheduler deconstructor

Parameters

p_ums_scheduler	NON-NULL pointer to the object to destroy

Definition at line 83 of file ums_scheduler.h.

4.13.2.2 DESTROY_UMS_SCHEDULER_SL

```
#define DESTROY_UMS_SCHEDULER_SL(
              p_ums_scheduler_sl )
```

Value:

```
    do{ \
        (p_ums_scheduler_sl)->key = 0; \
        (p_ums_scheduler_sl)->ums_scheduler = NULL;    \
        \
        (p_ums_scheduler_sl)->proc_entry = NULL;    \
        (p_ums_scheduler_sl)->proc_entry_info = NULL;    \
        (p_ums_scheduler_sl)->proc_entry_main_workers = NULL;   \
    } while(0)
```

ums_scheduler_sl destructor

Parameters

p_ums_scheduler↩_sl	NON-NULL pointer ums_scheduler_sl object to destroy

Definition at line 292 of file ums_scheduler.h.

4.13.2.3 INIT_UMS_SCHEDULER

```
#define INIT_UMS_SCHEDULER(
              p_ums_scheduler,
              p_scheduler_task_struct_in,
              p_completion_list_in )
```

Value:

```
    do{ \
        (p_ums_scheduler)->scheduler_task_struct = p_scheduler_task_struct_in;   \
        \
        (p_ums_scheduler)->completion_list = p_completion_list_in;   \
        (p_ums_scheduler)->current_completion_list_item = NULL; \
        \
        INIT_LIST_HEAD(&(p_ums_scheduler)->ready_list); \
        (p_ums_scheduler)->current_ready_list_item = NULL;   \
        \
        (p_ums_scheduler)->running_thread = NULL;    \
        (p_ums_scheduler)->num_switch = 0;    \
        (p_ums_scheduler)->cpu_core = -1;    \
    }while(0)
```

ums_scheduler constructor

Parameters

p_ums_scheduler	NON-NULL pointer to the object to init
p_scheduler_task_struct↩ _in	pointer to task_struct of the scheduler thread
p_completion_list_in	pointer to the ums_completion_list_sl to manage

Definition at line 63 of file ums_scheduler.h.

4.13.2.4 INIT_UMS_SCHEDULER_SL

```
#define INIT_UMS_SCHEDULER_SL(
                p_ums_scheduler_sl,
                key_in,
                p_ums_scheduler_in )
```

Value:

```
    do{ \
        (p_ums_scheduler_sl)->key = key_in; \
        spin_lock_init(&(p_ums_scheduler_sl)->ums_scheduler_spin_lock);  \
        \
        (p_ums_scheduler_sl)->ums_scheduler = p_ums_scheduler_in;   \
        \
        (p_ums_scheduler_sl)->proc_entry = NULL;   \
        (p_ums_scheduler_sl)->proc_entry_info = NULL;   \
        (p_ums_scheduler_sl)->proc_entry_main_workers = NULL;  \
    } while(0)
```

ums_scheduler_sl constructor

Parameters

p_ums_scheduler↩ _sl	NON-NULL pointer ums_scheduler_sl object to init
key_in	key in the hashtable of process' scheduler, corresponds to scheudler's pid
p_ums_scheduler↩ _in	NON-NULL pointer to the ums_scheduler to manage

Definition at line 275 of file ums_scheduler.h.

4.13.2.5 PRINTK_UMS_SCHEDULER

```
#define PRINTK_UMS_SCHEDULER(
                p_obj,
                PREFIX )
```

Value:

```
    do{ \
        char* __buff =  kmalloc(4096, GFP_KERNEL);  \
        if(snprintf_ums_scheduler(__buff, 4096, p_obj) > 4096-4)    \
            printk(KERN_DEBUG "\n PRINTK_UMS_SCHEDULER() overflow!!!\n");   \
        else printk(PREFIX "ums_scheduler_t = \n{\n%s}\n", __buff);    \
        kfree(__buff);  \
    }while(0)
```

printK a ums_scheduler

Definition at line 478 of file ums_scheduler.h.

4.13.2.6 PRINTK_UMS_SCHEDULER_SL

```
#define PRINTK_UMS_SCHEDULER_SL(
              p_obj,
              PREFIX )
```

Value:

```
    do{ \
        char* __buff =  kmalloc(4096, GFP_KERNEL);  \
        if( snprintf_ums_scheduler_sl(__buff, 4096, p_obj) > 4096-4)    \
            printk(KERN_DEBUG "\n PRINTK_UMS_SCHEDULER_SL() overflow!!!\n");   \
        else printk(PREFIX "ums_scheduler_sl_t = \n{\n%s}\n", __buff);    \
        kfree(__buff);  \
    }while(0)
```

printK ums_scheduler_sl

Definition at line 519 of file ums_scheduler.h.

4.13.2.7 ums_scheduler_completion_list_iterate

```
#define ums_scheduler_completion_list_iterate(
              p_ums_scheduler,
              p_ums_completion_list_item_out )
```

Value:

```
    do{ \
        (p_ums_scheduler)->current_completion_list_item =
       (p_ums_scheduler)->current_completion_list_item->next;    \
        if(likely((p_ums_scheduler)->current_completion_list_item !=
       &((p_ums_scheduler)->completion_list->ums_context_list)))  \
            p_ums_completion_list_item_out = list_entry((p_ums_scheduler)->current_completion_list_item,
       ums_completion_list_item_t, list);    \
        else p_ums_completion_list_item_out = NULL; \
    }while(0)
```

get next ums_completion_list_item during navigation

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_completion_list_item_out	output, pointer to ums_completion_item, return null at the end of the list

Definition at line 175 of file ums_scheduler.h.

4.13.2.8 ums_scheduler_completion_list_iterate_end

```
#define ums_scheduler_completion_list_iterate_end(
              p_ums_scheduler )
```

Value:

```
    do{ \
        (p_ums_scheduler)->current_completion_list_item = NULL; \
        spin_unlock(&((p_ums_scheduler)->completion_list->ums_context_list_spin_lock)); \
    }while(0)
```

end to iterate the completion_list

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler

Definition at line 189 of file ums_scheduler.h.

4.13.2.9 ums_scheduler_completion_list_iteration_get_current

```
#define ums_scheduler_completion_list_iteration_get_current(
              p_ums_scheduler,
              p_ums_completion_list_item_out )
```

Value:

```
    do{ \
        p_ums_completion_list_item_out = (p_ums_scheduler)->current_completion_list_item;   \
    }while(0)
```

get current element during the iteration of the completion_list

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_completion_list_item_out	output, pointer to ums_completion_item

Definition at line 162 of file ums_scheduler.h.

4.13.2.10 ums_scheduler_completion_list_start_iteration

```
#define ums_scheduler_completion_list_start_iteration(
              p_ums_scheduler,
              p_ums_completion_list_item_out )
```

Value:

```
    do{ \
        spin_lock(&((p_ums_scheduler)->completion_list->ums_context_list_spin_lock));    \
        (p_ums_scheduler)->current_completion_list_item =
       (p_ums_scheduler)->completion_list->ums_context_list.next;    \
        if(unlikely(list_empty(&((p_ums_scheduler)->completion_list->ums_context_list)))) \
            p_ums_completion_list_item_out = NULL;  \
        else    \
            p_ums_completion_list_item_out = list_entry((p_ums_scheduler)->current_completion_list_item,
       ums_completion_list_item_t, list);    \
    }while(0)
```

start to iterate the completion_list

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_completion_list_item_out	output, pointer to ums_completion_item. it's NULL if the list ends

Definition at line 145 of file ums_scheduler.h.

4.13.2.11 ums_scheduler_list_empty

```
#define ums_scheduler_list_empty(
             p_list_head ) list_empty(p_list_head)
```

macro used to check if a list is empty

Parameters

p_list_head	it can be the ready_list or the ums_context_list (ums_completion_list)

Definition at line 350 of file ums_scheduler.h.

4.13.2.12 ums_scheduler_ready_list_add

```
#define ums_scheduler_ready_list_add(
             p_ums_scheduler,
             p_ums_context )
```

Value:

```
    do{ \
        list_add_tail(&((p_ums_context)->list), &((p_ums_scheduler)->ready_list));  \
    }while(0)
```

add a ums_context to ready list of the scheduler

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_context	NON-NULL pointer to the ums_context to add

Definition at line 105 of file ums_scheduler.h.

4.13.2.13 ums_scheduler_ready_list_iterate

```
#define ums_scheduler_ready_list_iterate(
              p_ums_scheduler,
              p_ums_context_out )
```

Value:
```
    do{ \
        (p_ums_scheduler)->current_ready_list_item = (p_ums_scheduler)->current_ready_list_item->next;    \
        if(likely((p_ums_scheduler)->current_ready_list_item != &((p_ums_scheduler)->ready_list)))  \
            p_ums_context_out = list_entry((p_ums_scheduler)->current_ready_list_item, ums_context_t, list);
          \
        else p_ums_context_out = NULL; \
    }while(0)
```

get next ums_context during navigation

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_completion_list_item_out	output, pointer to ums_context

Definition at line 217 of file ums_scheduler.h.

4.13.2.14 ums_scheduler_ready_list_iterate_end

```
#define ums_scheduler_ready_list_iterate_end(
              p_ums_scheduler )
```

Value:
```
    do{ \
        (p_ums_scheduler)->current_ready_list_item = NULL; \
    }while(0)
```

end to iterate the ready_list

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler

Definition at line 231 of file ums_scheduler.h.

4.13.2.15 ums_scheduler_ready_list_remove

```
#define ums_scheduler_ready_list_remove(
              p_ums_scheduler,
              p_ums_context )
```

Value:
```
    do{ \
        list_del(&((p_ums_context)->list));  \
    }while(0)
```

remove a ums_context from the ready list of the scheduler

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_context	NON-NULL pointer to the ums_context to remove

Definition at line 116 of file ums_scheduler.h.

4.13.2.16 ums_scheduler_ready_list_remove_first

```
#define ums_scheduler_ready_list_remove_first(
              p_ums_scheduler,
              p_ums_context_OUT )
```

Value:

```
    do{ \
        p_ums_context_OUT = list_first_entry_or_null(&(p_ums_scheduler)->ready_list, ums_context_t, list);
        \
        if(likely((p_ums_context_OUT) != NULL))    \
            list_del(&((p_ums_context_OUT)->list));   \
    }while(0)
```

remove first ums_context from the ready list

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_context_OUT	output, pointer to a ums_context

Definition at line 129 of file ums_scheduler.h.

4.13.2.17 ums_scheduler_ready_list_start_iteration

```
#define ums_scheduler_ready_list_start_iteration(
              p_ums_scheduler,
              p_ums_context_out )
```

Value:

```
    do{ \
        (p_ums_scheduler)->current_ready_list_item = (p_ums_scheduler)->ready_list.next;     \
        p_ums_context_out = list_entry((p_ums_scheduler)->current_ready_list_item, ums_context_t, list);
       \
    }while(0)
```

start to iterate the ready_list

Parameters

p_ums_scheduler	NON-NULL pointer to the scheduler
p_ums_completion_list_item_out	output, pointer to ums_context

Definition at line 204 of file ums_scheduler.h.

4.13.2.18 ums_scheduler_set_reason_end_sheduler

```
#define ums_scheduler_set_reason_end_sheduler(
              p_ums_scheduler )
```

Value:
```
    do{ \
        (p_ums_scheduler)->entry_point_args->reason = REASON_SPECIAL_END_SCHEDULER; \
    }while(0)
```

set reason of the next scheduler call

Definition at line 242 of file ums_scheduler.h.

4.13.2.19 ums_scheduler_sl_lock_get_scheduler

```
#define ums_scheduler_sl_lock_get_scheduler(
              p_ums_scheduler_sl,
              p_ums_scheduler_OUT )
```

Value:
```
    do{ \
        spin_lock(&((p_ums_scheduler_sl)->ums_scheduler_spin_lock));    \
        p_ums_scheduler_OUT = (p_ums_scheduler_sl)->ums_scheduler;   \
    }while(0)
```

lock the ums_scheduler in the ums_scheduler_sl object

Parameters

p_ums_scheduler_sl	NON-NULL pointer ums_scheduler_sl object
p_ums_scheduler_OUT	output, pointer to the ums_scheduler locked

Definition at line 327 of file ums_scheduler.h.

4.13.2.20 ums_scheduler_sl_remove_scheduler

```
#define ums_scheduler_sl_remove_scheduler(
              p_ums_scheduler_sl,
              p_ums_scheduler_OUT )
```

Value:
```
    do{ \
        spin_lock(&((p_ums_scheduler_sl)->ums_scheduler_spin_lock));    \
        p_ums_scheduler_OUT = (p_ums_scheduler_sl)->ums_scheduler;   \
        (p_ums_scheduler_sl)->ums_scheduler = NULL;   \
        spin_unlock(&((p_ums_scheduler_sl)->ums_scheduler_spin_lock));    \
    }while(0)
```

remove ums_scheduler from the ums_scheduler_sl

Parameters

p_ums_scheduler_sl	NON-NULL pointer ums_scheduler_sl object
p_ums_scheduler_OUT	output, pointer to the ums_scheduler removed

Definition at line 311 of file ums_scheduler.h.

4.13.2.21 ums_scheduler_sl_unlock_scheduler

```
#define ums_scheduler_sl_unlock_scheduler(
             p_ums_scheduler_sl )
```

Value:

```
    do{ \
        spin_unlock(&((p_ums_scheduler_sl)->ums_scheduler_spin_lock));  \
    }while(0)
```

unlock the ums_scheduler in the ums_scheduler_sl object

Parameters

p_ums_scheduler↵_sl	NON-NULL pointer ums_scheduler_sl object

Definition at line 338 of file ums_scheduler.h.

4.13.3 Typedef Documentation

4.13.3.1 ums_scheduler_sl_t

```
typedef struct ums_scheduler_sl_t ums_scheduler_sl_t
```

object used to arrange a ums_scheduler in a hashtable and to protect it with a spin_lock

4.13.3.2 ums_scheduler_t

```
typedef struct ums_scheduler_t ums_scheduler_t
```

object that represent a ums_scheduler

Index