Simple Memory Manager (smm)

1.0

Generated by Doxygen 1.5.6

Fri Feb 6 22:50:50 2009

Contents

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 Data Structure Documentation **5**

3.1 info_smm_mem Struct Reference 5

3.1.1 Detailed Description 5

3.1.2 Field Documentation 5

3.1.2.1 max_hole 5

3.1.2.2 empty_space 5

3.1.2.3 holes 5

3.1.2.4 processes 5

3.2 packet Struct Reference 7

3.2.1 Detailed Description 7

3.2.2 Field Documentation 7

3.2.2.1 p_error 7

3.2.2.2 pid 7

3.2.2.3 size 7

3.2.2.4 size_type 7

3.2.2.5 address 7

3.2.2.6 cmd 7

3.3 smm_blk Struct Reference 8

3.3.1 Detailed Description 8

3.3.2 Field Documentation 8

3.3.2.1 start 8

3.3.2.2 last 8

3.3.2.3 swap 8

4 File Documentation **9**

4.1 inc/smm.h File Reference . . . 9

4.1.1 Function Documentation . . . 9

4.1.1.1 lock_mutex . . . 9

4.1.1.2 smm_calloc . . . 10

4.1.1.3 smm_chk_ptr . . . 10

4.1.1.4 smm_free . . . 10

4.1.1.5 smm_get . . . 11

4.1.1.6 smm_malloc . . . 11

4.1.1.7 smm_put . . . 11

4.1.1.8 smm_realloc . . . 12

4.1.1.9 unlock_mutex . . . 12

4.2 inc/smmMem.h File Reference . . . 13

4.2.1 Define Documentation . . . 14

4.2.1.1 KB . . . 14

4.2.1.2 MB . . . 14

4.2.2 Typedef Documentation . . . 14

4.2.2.1 elem_list . . . 14

4.2.2.2 info_smm_mem . . . 14

4.2.3 Function Documentation . . . 14

4.2.3.1 allocator . . . 14

4.2.3.2 best_fit . . . 15

4.2.3.3 check_logical_addr . . . 15

4.2.3.4 check_physical_addr . . . 15

4.2.3.5 compact_smm_mem . . . 16

4.2.3.6 fusion_blocks . . . 16

4.2.3.7 get_log_start_addr . . . 16

4.2.3.8 get_next . . . 16

4.2.3.9 get_phy_start_addr . . . 17

4.2.3.10 get_pid . . . 17

4.2.3.11 get_size_block . . . 17

4.2.3.12 how_many_blocks . . . 18

4.2.3.13 info_smm_mem_blocks . . . 18

4.2.3.14 init_smm_mem_info . . . 18

4.2.3.15 insert_block . . . 18

4.2.3.16 log_msg_error . . . 19

4.2.3.17 logical_to_physical_addr . . . 19
4.2.3.18 new_elem_list . . . 19
4.2.3.19 remove_all_blocks . . . 19
4.2.3.20 remove_block . . . 20
4.2.3.21 search_block . . . 20
4.2.3.22 search_max_hole . . . 20
4.2.3.23 set_log_start_addr . . . 20
4.2.3.24 set_next . . . 21
4.2.3.25 set_phy_start_addr . . . 21
4.2.3.26 set_pid . . . 21
4.2.3.27 set_size_block . . . 21
4.2.3.28 smmd_alloc . . . 22
4.2.3.29 smmd_expand . . . 22
4.2.3.30 smmd_realloc . . . 22
4.2.3.31 smmd_resize . . . 23
4.2.3.32 subset_allocated_blocks . . . 23
4.2.3.33 update_smm_mem_info . . . 24
4.2.3.34 write_to_log . . . 24
4.2.4 Variable Documentation . . . 24
4.2.4.1 info . . . 24
4.2.4.2 smmMem . . . 24
4.3 inc/utility.h File Reference . . . 25
4.3.1 Define Documentation . . . 26
4.3.1.1 FIFOFROM . . . 26
4.3.1.2 FIFOTO . . . 26
4.3.1.3 FILEPID . . . 26
4.3.1.4 FILEPIDSHELL . . . 26
4.3.1.5 LINES . . . 26
4.3.1.6 MAX . . . 26
4.3.1.7 open_fifos . . . 26
4.3.2 Typedef Documentation . . . 26
4.3.2.1 command . . . 26
4.3.2.2 packet . . . 26
4.3.3 Enumeration Type Documentation . . . 27
4.3.3.1 command . . . 27
4.3.4 Function Documentation . . . 27

4.3.4.1 fill_packet 27
4.3.4.2 open_fifo 27
4.3.4.3 read_from 28
4.3.4.4 reset_vett_cmd 28
4.3.4.5 tokenizza 28
4.3.4.6 verifica_comando 28
4.3.4.7 verifica_comando_server 29
4.3.4.8 verifica_indirizzo_esadecimale 29
4.3.4.9 verifica_valore 29
4.3.4.10 write_to 29

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

info_smm_mem 5
packet 7
smm_blk 8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

inc/smm.h 9
inc/smmMem.h 13
inc/utility.h 25

Chapter 3

Data Structure Documentation

3.1 info_smm_mem Struct Reference

`#include <smmMem.h>`

Data Fields

- size_t max_hole
- size_t empty_space
- struct elem_list * holes
- struct elem_list * processes

3.1.1 Detailed Description

Definition at line 15 of file smmMem.h.

3.1.2 Field Documentation

3.1.2.1 size_t info_smm_mem::max_hole

Definition at line 17 of file smmMem.h.

3.1.2.2 size_t info_smm_mem::empty_space

Definition at line 18 of file smmMem.h.

3.1.2.3 struct elem_list* info_smm_mem::holes [read]

Definition at line 19 of file smmMem.h.

3.1.2.4 struct elem_list* info_smm_mem::processes [read]

Definition at line 20 of file smmMem.h.

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

- inc/smmMem.h

3.2 packet Struct Reference

`#include <utility.h>`

Data Fields

- int p_error
- pid_t pid
- size_t size
- size_t size_type
- unsigned int address
- char cmd [10]

3.2.1 Detailed Description

Definition at line 38 of file utility.h.

3.2.2 Field Documentation

3.2.2.1 int packet::p_error

Definition at line 40 of file utility.h.

3.2.2.2 pid_t packet::pid

Definition at line 41 of file utility.h.

3.2.2.3 size_t packet::size

Definition at line 42 of file utility.h.

3.2.2.4 size_t packet::size_type

Definition at line 43 of file utility.h.

3.2.2.5 unsigned int packet::address

Definition at line 44 of file utility.h.

3.2.2.6 char packet::cmd[10]

Definition at line 45 of file utility.h.

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

- inc/utility.h

3.3 smm_blk Struct Reference

`#include <smm.h>`

Data Fields

- void * start
- void * last
- char swap

3.3.1 Detailed Description

Definition at line 10 of file smm.h.

3.3.2 Field Documentation

3.3.2.1 void* smm_blk::start

Definition at line 12 of file smm.h.

3.3.2.2 void* smm_blk::last

Definition at line 13 of file smm.h.

3.3.2.3 char smm_blk::swap

Definition at line 14 of file smm.h.

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

- inc/smm.h

Chapter 4

File Documentation

4.1 inc/smm.h File Reference

```
#include <sys/types.h>
```

Data Structures

- struct smm_blk

Functions

- void * smm_malloc (size_t size)
- void * smm_calloc (size_t nmemb, size_t size)
- void * smm_realloc (void *ptr, size_t size)
- int smm_free (void *ptr)
- int smm_put (void *buf, void *ptr, size_t size)
- int smm_get (void *ptr, void *buf, size_t size)
- int smm_chk_ptr (void *ptr, struct smm_blk *blk)
- int lock_mutex (int fd)
- int unlock_mutex (int)

4.1.1 Function Documentation

4.1.1.1 int lock_mutex (int *fd*)

Funzione utilizzata per richiedere il lock sul file descriptor

Parameters:

fd,file descriptor del file sul quale richiedere il lock.

Returns:

0, in caso di successo.
< 0, in caso di errore

4.1.1.2 void* smm_calloc (size_t *nmemb*, size_t *size*)

Funzione che alloca un blocco in smmMem per un array di nmemb elementi di dimensione pari a size byute ognuno. La memoria allocata e' inizializzata a 0.

Precondition:

nmemb >0 && size > 0.

Parameters:

nmemb,numero elementi.

size,dimesione tipo dato.

Returns:

puntatore all'inizio del blocco.
NULL, se si e' verificato un errore.

4.1.1.3 int smm_chk_ptr (void * *ptr*, struct smm_blk * *blk*)

Funzione che rileva le informazione al blocco a cui appartiene ptr, riempiendo opportunamente i campi della struttura pre-allocata dal chiamante (ex: smmMon). Il campo swap della strttura viene impostato di default a 0 (swapped in).

Precondition:

ptr != NULL.
blk != NULL.

Parameters:

ptr,indirizzo da ricercare.

blk,blocco a cui appartiene ptr.

Returns:

0, in caso di successo.
< 0, in caso di errore.

4.1.1.4 int smm_free (void * *ptr*)

Funzione che libera il blocco di memoria in smmMem precedentemente allocato e puntato dal puntatore 'ptr'.

Precondition:

ptr != NULL.

Parameters:

ptr,puntatore del blocco da liberare.

Returns:

< 0 in caso di errore.
0 in caso di successo.

4.1.1.5 int smm_get (void * *ptr*, void * *buf*, size_t *size*)

Funzione che preleva a partire dal puntatore 'ptr' appartenente a un blocco di memoria precedentemente allocato in smmMem un numero di byte pari a 'size' e li inserisce nel buffer interno all'applicazione puntato da 'buf'.

Precondition:

ptr != NULL.
buf != NULL.
size > 0.

Parameters:

ptr,indirizzo del blocco.

buff,dati da scrivere.

size,numero dati da scrivere.

Returns:

0, in caso di successo.
< 0, in caso di errore.

4.1.1.6 void* smm_malloc (size_t *size*)

Funzione che alloca un blocco di memoria in smmMem di dimensione pari a size byte. La memoria non viene azzerata (wasted).

Precondition:

size > 0.

Parameters:

size,numero elementi.

Returns:

puntatore all'inizio del blocco allocato.
NULL, se si e' verificato un errore.

4.1.1.7 int smm_put (void * *buf*, void * *ptr*, size_t *size*)

Funzione che inserisce a partire dal puntatore 'ptr' appartenente a un blocco di memoria precedentemente allocato in smmMem un numero di bytes pari a 'size' copiandoli dal buffer interno dell'applicazione puntato da 'buf'.

Precondition:

buf != NULL.
ptr != NULL.
size > 0.

Parameters:

buf,dati da scrivere.

ptr,indirizzo del blocco.

size,numero dati da scrivere.

Returns:

0, in caso di successo.
< 0, in caso di errore.

4.1.1.8 void* smm_realloc (void * *ptr*, size_t *size*)

Funzione che cambia a 'size' byte la dimensione del blocco di memoria gia' allocato in smmMem e puntato da 'ptr'. Il contenuto rimane invariato per la porzione di memoria di dimensione pari al minimo tra la vecchia e la nuova dimensione; eventuale memoria in aggiunta risultera' non inizializzata. Se il blocco puntato richiede uno spostamento (puntatore ritornato != da 'ptr'), la funzione garantisce la liberazione del vecchio puntatore, ovvero, viene effettuata una free('ptr').

Precondition:

size > 0
ptr != NULL

Parameters:

ptr,puntatore del blocco da rilocare

size,dimensione del blocco

Returns:

'ptr' || un nuovo puntatore.
NULL, in caso di errore.

4.1.1.9 int unlock_mutex (int)

Funzione utilizzata per rilasciare il lock sul file descriptor

Parameters:

fd,file descriptor del file da rilasciare

Returns:

0, in caso di successo.
< 0, in caso di errore

4.2 inc/smmMem.h File Reference

```
#include <sys/types.h>
#include "utility.h"
```

Data Structures

- struct info_smm_mem

Defines

- #define KB 1024
- #define MB KB ∗ KB

Typedefs

- typedef struct elem_list elem_list
- typedef struct info_smm_mem info_smm_mem

Functions

- elem_list ∗ new_elem_list (void)
- void set_phy_start_addr (elem_list ∗blk, unsigned int physical_addr)
- void set_log_start_addr (elem_list ∗blk, unsigned int logical_addr)
- void set_size_block (elem_list ∗blk, size_t size)
- void set_pid (elem_list ∗blk, pid_t pid)
- void set_next (elem_list ∗blk, elem_list ∗next)
- unsigned int get_phy_start_addr (elem_list ∗blk)
- unsigned int get_log_start_addr (elem_list ∗blk)
- size_t get_size_block (elem_list ∗blk)
- pid_t get_pid (elem_list ∗blk)
- elem_list ∗ get_next (elem_list ∗blk)
- int init_smm_mem_info ()
- void update_smm_mem_info (size_t size)
- elem_list ∗ insert_block (elem_list ∗head, unsigned int physical_addr, unsigned int logical_addr, pid_t pid, size_t size)
- elem_list ∗ search_block (elem_list ∗head, unsigned int address)
- int how_many_blocks (elem_list ∗head)
- elem_list ∗ remove_block (elem_list ∗head, unsigned int physical_addr)
- int remove_all_blocks (elem_list ∗∗head_ref)
- void info_smm_mem_blocks (elem_list ∗head, char ∗buff)
- elem_list ∗ subset_allocated_blocks (elem_list ∗head, pid_t pid)
- size_t search_max_hole (elem_list ∗head)
- elem_list ∗ fusion_blocks (elem_list ∗head, elem_list ∗blk_fusion)
- int allocator (elem_list ∗best_fit, size_t size, unsigned int ∗logical_addr, pid_t pid)
- int compact_smm_mem ()
- elem_list ∗ best_fit (elem_list ∗head, size_t size)
- int check_physical_addr (elem_list ∗head, unsigned int physical_addr)

- elem_list * check_logical_addr (elem_list *head, unsigned int logical_addr)
- unsigned int logical_to_physical_addr (elem_list *blk, unsigned int logical_addr)
- void log_msg_error (char msg[ ], char *func, int err)
- void write_to_log (int fd_server_log, char *buff_error, char *syscall, int error)
- int smmd_alloc (unsigned int *logical_address, packet *send_data)
- int smmd_resize (elem_list *blk, packet *send_data)
- int smmd_expand (elem_list *blk, packet *send_data, unsigned int *logical_addr)
- int smmd_realloc (elem_list *blk, packet *send_data, unsigned int *logical_addr)

Variables

- unsigned char * smmMem
- info_smm_mem * info

4.2.1 Define Documentation

4.2.1.1 #define KB 1024

Definition at line 7 of file smmMem.h.

4.2.1.2 #define MB KB * KB

Definition at line 8 of file smmMem.h.

4.2.2 Typedef Documentation

4.2.2.1 typedef struct elem_list elem_list

Definition at line 12 of file smmMem.h.

4.2.2.2 typedef struct info_smm_mem info_smm_mem

Definition at line 23 of file smmMem.h.

4.2.3 Function Documentation

4.2.3.1 int allocator (elem_list * *best_fit*, size_t *size*, unsigned int * *logical_addr*, pid_t *pid*)

Funzione che permette di inserire un nuovo blocco nella lista dei blocchi occupati:

1) invocazione della funzione insert_block() a cui, tra gli altri parametri viene passato il blocco calcolato attraverso l'invocazione della funzione best_fit().

2) viene ridimensionato o cancellato il blocco bestfit a seguito dell'inserimento.

3) viene calcolato il prossimo indirizzo logico da assegnare.

Parameters:

best_fit,blocco "migliore"

size,dimensione del blocco da allocare

logical_addr,indirizzo logical_addr da assegnare al nuovo blocco

pid,pid.

Returns:

testa della lista aggiornata.
< 0, in caso di errore.

4.2.3.2 elem_list* best_fit (elem_list * *head*, size_t *size*)

Funzione che implementa l'algoritmo di selezione dell'area (placement algorithm) di tipo best-fit. La funzione prende in input la lista dei blocchi liberi (info->holes) e ricerca il blocco piu' piccolo >= di size.

Parameters:

head,testa della lista.

size,dimensione da ricercare.

Returns:

puntatore, al blocco trovato.
NULL, blocco non trovato.

4.2.3.3 elem_list* check_logical_addr (elem_list * *head*, unsigned int *logical_addr*)

Funzione che presa in input una lista ed un indirizzo logico, verifica se tale indirizzo appartiene ad un range.

Parameters:

head,testa della lista.

logical_addr,indirizzo da verificare.

Returns:

blocco contiene l'indirizzo logico.
NULL, in caso di insuccesso.

4.2.3.4 int check_physical_addr (elem_list * *head*, unsigned int *physical_addr*)

Funzione che presa in input una lista ed un indirizzo fisico, verifica se tale indirizzo appartiene ad un range.

Parameters:

head,testa della lista.

physical_addr,indirizzo da verificare.

Returns:

0, in caso di successo.
-1 , in caso di insuccesso.

4.2.3.5 int compact_smm_mem ()

Funzione che permette di compattare la lista dei blocchi occupati (info->processes):

1) per ogni blocco in info->processes, viene decrementato l'indirizzo fisico di un offset opportunamente calcolato.

2) vengono spostati in smmMem, i dati relativi al blocco preso in esame.

Returns:

0.

4.2.3.6 elem_list* fusion_blocks (elem_list * *head*, elem_list * *blk_fusion*)

Funzione che inserisce un blocco rimosso dalla lista dei blocchi occupati (info>processes), nella lista dei blocchi liberi (info->holes). L'inserimento puo' avviene confrontando l'indirizzo fisico del blocco da inserire con l'indirizzo fisico del blocco della lista preso in esame:

1) il blocco può essere inserito 2) fuso con il suo successire 3) oppure fuso con il suo predecessore ed eventualmente anche con il suo successore.

Parameters:

head,testa della lista.

blk_fusion

Returns:

testa della lista aggiornata.

4.2.3.7 unsigned int get_log_start_addr (elem_list * *blk*)

Funzione che ritorna l'indirizzo logico del blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

Returns:

indirizzo logico.

4.2.3.8 elem_list* get_next (elem_list * *blk*)

Funzione che ritorna il next (blocco successivo) del blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

Returns:

next (blocco successivo).

4.2.3.9 unsigned int get_phy_start_addr (elem_list * *blk*)

Funzione che ritorna l'indirizzo fisico del blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

Returns:

indirizzo fisico.

4.2.3.10 pid_t get_pid (elem_list * *blk*)

Funzione che ritorna il pid del blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

Returns:

pid.

4.2.3.11 size_t get_size_block (elem_list * *blk*)

Funzione che ritorna la dimensione del blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

Returns:

dimensione del blocco.

4.2.3.12 int how_many_blocks (elem_list * *head*)

Funzione che ritorna il numero di blocchi appartenenti ad una lista.

Parameters:

head,testa della lista.

Returns:

numero di blocchi.

4.2.3.13 void info_smm_mem_blocks (elem_list * *head*, char * *buff*)

Funzione usata a seguito del comando lab/lfb di smmMon, che copia nel buff passato come parametro le informazioni contenute nella lista. Questo viene fatto attraverso l'uso della funzione snprintf(...). Nel caso del comando lfb la sintassi usata e' la seguente:

snprintf(buff, 40, "%p %p %d\n",phy_start, phy_last, size);

nel caso del comando lab o lab [PID] la sintassi usata e' la seguente:

snprintf(buff, 40, "%d %p %p %d\n", PID, phy_start, phy_last, size).

Precondition:

buff != NULL.

Parameters:

head,testa della lista.

buff,buffer contenente i dati scritti.

4.2.3.14 int init_smm_mem_info ()

Funzione che inizializza le strutture utilizzate da smmd per gestire la memoria e crea le FIFO per la comunicazione con i clients.

Returns:

0, in caso di successo.
errno, in caso di errore.

4.2.3.15 elem_list* insert_block (elem_list * *head*, unsigned int *physical_addr*, unsigned int *logical_addr*, pid_t *pid*, size_t *size*)

Funzione per inserire un blocco in una lista ordinata per indirizzi fisici. All'interno viene invocata la funzione check_physical_addr(...) per verificare che il blocco nuovo da inserire non vada a sovrascrivere un blocco precedentemente allocato.

Parameters:

head,testa della lista.

physical_addr,indirizzo fisico.
logical_addr,indirizzo logico.
pid,pid.
size,dimensione.

Returns:

testa della lista aggiornata.
NULL, in caso di errore

4.2.3.16 void log_msg_error (char *msg*[ ], char * *func*, int *err*)

Funzione che crea un messaggio di errore da scrivere nel file smmd.log del server.

Parameters:

msg,messaggio da scrivere.
func,nome della syscall da scrivere.
err,errore da scrivere.

4.2.3.17 unsigned int logical_to_physical_addr (elem_list * *blk*, unsigned int *logical_addr*)

Funzione che mappa un indirizzo logico con un indirizzo fisico corrispondente.

Parameters:

blk,blocco contenente l'indirizzo fisico.
logical_addr,indirizzo logico.

Returns:

indirizzo fisico.

4.2.3.18 elem_list* new_elem_list (void)

Costruttore per il tipo di dato puntatore elem_list (blocco).

Returns:

nuovo elemento elem_list.
NULL, in caso di errore da parte della syscall malloc().

4.2.3.19 int remove_all_blocks (elem_list ** *head_ref*)

Funzione che elimina tutti i blocchi appartenenti alla lista passata come parametro.

Parameters:

head,testa della lista.

Returns:

0.

4.2.3.20 elem_list* remove_block (elem_list * *head*, unsigned int *physical_addr*)

Funzione che data una lista ed un indirizzo fisico, ricerca e successivamente rimuove il blocco a cui quell'indirizzo fisico appartiene.

Parameters:

head,testa della lista.
physical_addr,indirizzo fisico da ricercare.

Returns:

testa della lista aggiornata.

4.2.3.21 elem_list* search_block (elem_list * *head*, unsigned int *address*)

Funzione che dato un indirizzo, ricerca il blocco a cui questo appartiene.

Parameters:

head,testa della lista.
address,indirizzo da ricercare.

Returns:

blk, contenente l'indirizzo.
NULL, in caso di insuccesso.

4.2.3.22 size_t search_max_hole (elem_list * *head*)

Funzione che presa in input la lista dei blocchi liberi (info->holes), ritorna il blocco di dimensione massima.

Parameters:

head,testa della lista.

Returns:

0, se la lista e' vuota.
dimensione massima

4.2.3.23 void set_log_start_addr (elem_list * *blk*, unsigned int *logical_addr*)

Funzione che assegna l'indirizzo logico al blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.
logical_address,indirizzo logico da assegnare.

4.2.3.24 void set_next (elem_list * *blk*, elem_list * *next*)

Funzione che assegna il next (blocco successivo) di un blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

next,(blocco successivo) da assegnare.

4.2.3.25 void set_phy_start_addr (elem_list * *blk*, unsigned int *physical_addr*)

Funzione che assegna l'indirizzo fisico al blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

physical_addr,indirizzo fisico da assegnare.

4.2.3.26 void set_pid (elem_list * *blk*, pid_t *pid*)

Funzione che assegna il pid al blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

pid,pid da assegnare.

4.2.3.27 void set_size_block (elem_list * *blk*, size_t *size*)

Funzione che assegna la dimensione al blocco.

Precondition:

blk != NULL.

Parameters:

blk,blocco considerato.

size,dimensione da assegnare.

4.2.3.28 int smmd_alloc (unsigned int * *logical_address*, packet * *send_data*)

Funzione che viene eseguita a seguito di una richiesta di smm_malloc/smm_calloc:

1) verifica dello spazio disponibile per poter soddisfare la richiesta.

2) se la richiesta puo' essere soddisfatta, viene eseguito un ulteriore controllo per vericare se e' necessario eseguire compattazione.

3) viene chiamata la funzione allocator(...)

4) nel caso in cui la richiesta si smm_calloc(), la memoria assegnata al processo viene inizializzata a 0.

Parameters:

logical_address,indirizzo logico da assegnare.

send_data,richiesta.

Returns:

0, in caso di successo.
-1, in caso di errore syscall malloc().

4.2.3.29 int smmd_expand (elem_list * *blk*, packet * *send_data*, unsigned int * *logical_addr*)

Funzione che permette di espandere il blocco 'blk' a seguito di una richiesta di smm_realloc(). Il procedimento avviene nel seguente modo:

1) ricerca della hole adiacente del blocco da riallocare.

2) se la hole trovata, soddisfa la richiesta:

2.a) vengono opportunamente aggiornate le informazioni del blocco da riallocare (dimensione).

2.b) aggiornamento campi della hole o rimozione della hole.

3) se la hole trovata, non soddisfa la richiesta: se il blocco da riallocare e' l'ultimo blocco della lista dei blocchi occupati, viene effettuata compattazione e aggiornti i campi della e le informazioni relative ad smmMem; altrimenti viene invocata la funzione smmd_realloc()

Parameters:

blk,blocco da riallocare

send_data,richiesta.

logical_addr,indirizzo logico da assegnare

Returns:

0, in caso di successo.
-1, in caso di errore syscall malloc().

4.2.3.30 int smmd_realloc (elem_list * *blk*, packet * *send_data*, unsigned int * *logical_addr*)

Funzione che permette di spostare fisicamente il blocco 'blk' da una zona di smmMem ad un' altra spostando opportunamente anche i suoi dati.

1) viene effettuata una copia dei dati temporanea.

2) viene creata una copia del blocco da riallocare.

3) viene rimosso il blocco da riallocare.

4) viene inserito nuovamente il blocco

5) vengono ricopiati i dati in smmMem nella sua nuova locazione.

Parameters:

blk,blocco da riallocare.

send_data,richiesta.

logical_addr,indirizzo logico da assegnare.

Returns:

0, in caso di successo.
-1, in caso di errore syscall malloc().

4.2.3.31 int smmd_resize (elem_list * *blk*, packet * *send_data*)

Funzione invocata a seguito di una richiesta di smm_realloc(). La permette di:

1) ridimensionare il blocco 'blk', aggiornando la sua dimensione alla nuova dimensione.

2) aggiornare la lista dei blocchi liberi, attraverso l'invocazione della funzione fusion().

Parameters:

blk,blocco da ridimensionare

send_data,richiesta

Returns:

0, in caso di successo.
-1, in caso di errore syscall malloc().

4.2.3.32 elem_list* subset_allocated_blocks (elem_list * *head*, pid_t *pid*)

Funzione che presa in input una lista e un pid, crea una sottolista contenente il PID richiesto.

Parameters:

head,testa della lista.

pid,pid richiesto.

Returns:

sottolista, in caso di successo.
NULL, in caso di errore.

4.2.3.33 void update_smm_mem_info (size_t *size*)

Funzione che aggiorna i campi empty_space e max_hole dell' handler di memoria (info).

Parameters:

size,dimensione da aggiornare.

4.2.3.34 void write_to_log (int *fd_server_log*, char * *buff_error*, char * *syscall*, int *error*)

Funzione che scrive sul file smmd.log il messaggio di errore, precedentemente creato dalla funzione log_-msg_error(...).

Parameters:

fd_server_log,file descriptor di smmd.log

buff_error,messaggio di errore

syscall,parametro func di log_msg_error(...).

error,errore da settare.

4.2.4 Variable Documentation

4.2.4.1 info_smm_mem* info

4.2.4.2 unsigned char* smmMem

4.3 inc/utility.h File Reference

```
#include <sys/types.h>
```

Data Structures

- struct packet

Defines

- #define MAX 256
- #define LINES 2
- #define FIFOTO "/tmp/FIFOtosmmd"
- #define FIFOFROM "/tmp/FIFOfromsmmd"
- #define FILEPID "/tmp/smmd.pid"
- #define FILEPIDSHELL "/tmp/smmMon.pid"
- #define open_fifos(fdto, fdfrom, ret)

Typedefs

- typedef enum command command
- typedef struct packet packet

Enumerations

- enum command {
  sm_start, sm_stop, sm_exit, sm_alloc,
  sm_free, sm_check, sm_lab, sm_lfb,
  sm_not_found, sm_malloc, sm_calloc, sm_realloc,
  sm_put, sm_get }

Functions

- int tokenizza (char *command, char *array_cmd[ ])
- void reset_vett_cmd (char *array_cmd[ ])
- command verifica_comando (char *array_cmd[ ])
- int verifica_valore (char *array_cmd[ ])
- int verifica_indirizzo_esadecimale (char *array_cmd[ ])
- command verifica_comando_server (char *cmd)
- void fill_packet (packet *p, size_t size, size_t type, void *address, char *cmd)
- int write_to (int fd_fifo, const void *buff, int w_bytes)
- int read_from (int fd_fifo, const void *buff, int r_bytes)
- int open_fifo (const char *fifo_name)

4.3.1 Define Documentation

4.3.1.1 #define FIFOFROM "/tmp/FIFOfromsmmd"

Definition at line 10 of file utility.h.

4.3.1.2 #define FIFOTO "/tmp/FIFOtosmmd"

Definition at line 9 of file utility.h.

4.3.1.3 #define FILEPID "/tmp/smmd.pid"

Definition at line 11 of file utility.h.

4.3.1.4 #define FILEPIDSHELL "/tmp/smmMon.pid"

Definition at line 12 of file utility.h.

4.3.1.5 #define LINES 2

Definition at line 7 of file utility.h.

4.3.1.6 #define MAX 256

Definition at line 6 of file utility.h.

4.3.1.7 #define open_fifos(fdto, fdfrom, ret)

Value:

```
do {                                    \
          fdto = open_fifo(FIFOTO);             \
          fdfrom = open_fifo(FIFOFROM);         \
                                                \
          if((fdto < 0) || (fdfrom < 0)) {      \
               return (ret);                    \
          }                                     \
     } while(0)
```

Definition at line 185 of file utility.h.

4.3.2 Typedef Documentation

4.3.2.1 typedef enum command command

Definition at line 33 of file utility.h.

4.3.2.2 typedef struct packet packet

Definition at line 48 of file utility.h.

4.3.3 Enumeration Type Documentation

4.3.3.1 enum command

Enumerator:

sm_start
sm_stop
sm_exit
sm_alloc
sm_free
sm_check
sm_lab
sm_lfb
sm_not_found
sm_malloc
sm_calloc
sm_realloc
sm_put
sm_get

Definition at line 16 of file utility.h.

4.3.4 Function Documentation

4.3.4.1 void fill_packet (packet * *p*, size_t *size*, size_t *type*, void * *address*, char * *cmd*)

Funzione che permette di inizializzare i campi di un packet.

Parameters:

p,puntatore al packet.
size,dimensione.
type,dimensione del tipo di dato.
addrres,indirizzo.
cmd,comando da eseguire.

4.3.4.2 int open_fifo (const char * *fifo_name*)

Funzione che permette di aprire una fifo.

Parameters:

fifo_name,path.

Returns:

0, in caso di successo.
-1, in caso di errore.

4.3.4.3 int read_from (int *fd_fifo*, const void * *buff*, int *r_bytes*)

Wrapper della syscall read.

Parameters:

fd_fifo,file descriptor.

buff,dati letti.

r_bytes,numero dei dati letti.

Returns:

0, in caso di successo.
errno, in caso di errore

4.3.4.4 void reset_vett_cmd (char * *array_cmd*[ ])

Funzione che permette di deallocare il vettore di puntatori di stringhe.

Parameters:

array_cmd,array di stringhe

4.3.4.5 int tokenizza (char * *command*, char * *array_cmd*[ ])

Funzione che permette attraverso ripetute chiamate di strtok() di suddividere una stringa in token e di memorizzarli in un vettore di puntatori di stringhe.

Parameters:

command,comando da analizzare.

array_cmd,array di stringhe.

Returns:

<0, in caso di errore.
0, in caso di successo.

4.3.4.6 command verifica_comando (char * *array_cmd*[ ])

Funzione che verifica la correttezza di un comando (lato smmMon).

Parameters:

array_cmd,array di stringe.

Returns:

command, comando ritornato.

4.3.4.7 command verifica_comando_server (char * *cmd*)

Funzione che verifica la correttezza di un comando (lato smmd).

Parameters:

cmd,stringa del comando.

Returns:

command, comando ritornato.

4.3.4.8 int verifica_indirizzo_esadecimale (char * *array_cmd*[ ])

Funzione che verifica se una stringa è in formato esadecimale.

Parameters:

array_cmd,array di stringhe.

Returns:

0, in caso di successo.
<0, in caso di errore.

4.3.4.9 int verifica_valore (char * *array_cmd*[ ])

Funzione che verifica se una stringa passata come argometo è composta da soli numeri.

Parameters:

array_cmd,array di stringhe.

Returns:

0, in caso di successo.
< 0, in caso di errore.

4.3.4.10 int write_to (int *fd_fifo*, const void * *buff*, int *w_bytes*)

Wrapper della syscall write.

param fd_fifo, file descriptor.

Parameters:

buff,dati da scrivere.

w_bytes,numero dei dati da scrivere.

Returns:

0, in caso di successo.
errno, in caso di errore

Index

address
    packet, 7
allocator
    smmMem.h, 14

best_fit
    smmMem.h, 15

check_logical_addr
    smmMem.h, 15
check_physical_addr
    smmMem.h, 15
cmd
    packet, 7
command
    utility.h, 26, 27
compact_smm_mem
    smmMem.h, 15

elem_list
    smmMem.h, 14
empty_space
    info_smm_mem, 5

FIFOFROM
    utility.h, 26
FIFOTO
    utility.h, 26
FILEPID
    utility.h, 26
FILEPIDSHELL
    utility.h, 26
fill_packet
    utility.h, 27
fusion_blocks
    smmMem.h, 16

get_log_start_addr
    smmMem.h, 16
get_next
    smmMem.h, 16
get_phy_start_addr
    smmMem.h, 17
get_pid
    smmMem.h, 17
get_size_block
    smmMem.h, 17

holes
    info_smm_mem, 5
how_many_blocks
    smmMem.h, 17

inc/smm.h, 9
inc/smmMem.h, 13
inc/utility.h, 25
info
    smmMem.h, 24
info_smm_mem, 5
    empty_space, 5
    holes, 5
    max_hole, 5
    processes, 5
    smmMem.h, 14
info_smm_mem_blocks
    smmMem.h, 18
init_smm_mem_info
    smmMem.h, 18
insert_block
    smmMem.h, 18

KB
    smmMem.h, 14

last
    smm_blk, 8
LINES
    utility.h, 26
lock_mutex
    smm.h, 9
log_msg_error
    smmMem.h, 19
logical_to_physical_addr
    smmMem.h, 19

MAX
    utility.h, 26
max_hole
    info_smm_mem, 5
MB
    smmMem.h, 14

new_elem_list
smmMem.h, 19

open_fifo
utility.h, 27
open_fifos
utility.h, 26

p_error
packet, 7
packet, 7
address, 7
cmd, 7
p_error, 7
pid, 7
size, 7
size_type, 7
utility.h, 26
pid
packet, 7
processes
info_smm_mem, 5

read_from
utility.h, 27
remove_all_blocks
smmMem.h, 19
remove_block
smmMem.h, 19
reset_vett_cmd
utility.h, 28

search_block
smmMem.h, 20
search_max_hole
smmMem.h, 20
set_log_start_addr
smmMem.h, 20
set_next
smmMem.h, 20
set_phy_start_addr
smmMem.h, 21
set_pid
smmMem.h, 21
set_size_block
smmMem.h, 21
size
packet, 7
size_type
packet, 7
sm_alloc
utility.h, 27
sm_calloc
utility.h, 27
sm_check
utility.h, 27
sm_exit
utility.h, 27
sm_free
utility.h, 27
sm_get
utility.h, 27
sm_lab
utility.h, 27
sm_lfb
utility.h, 27
sm_malloc
utility.h, 27
sm_not_found
utility.h, 27
sm_put
utility.h, 27
sm_realloc
utility.h, 27
sm_start
utility.h, 27
sm_stop
utility.h, 27
smm.h
lock_mutex, 9
smm_calloc, 9
smm_chk_ptr, 10
smm_free, 10
smm_get, 10
smm_malloc, 11
smm_put, 11
smm_realloc, 12
unlock_mutex, 12
smm_blk, 8
last, 8
start, 8
swap, 8
smm_calloc
smm.h, 9
smm_chk_ptr
smm.h, 10
smm_free
smm.h, 10
smm_get
smm.h, 10
smm_malloc
smm.h, 11
smm_put
smm.h, 11
smm_realloc
smm.h, 12
smmd_alloc
smmMem.h, 21

smmd_expand
smmMem.h, 22
smmd_realloc
smmMem.h, 22
smmd_resize
smmMem.h, 23
smmMem
smmMem.h, 24
smmMem.h
allocator, 14
best_fit, 15
check_logical_addr, 15
check_physical_addr, 15
compact_smm_mem, 15
elem_list, 14
fusion_blocks, 16
get_log_start_addr, 16
get_next, 16
get_phy_start_addr, 17
get_pid, 17
get_size_block, 17
how_many_blocks, 17
info, 24
info_smm_mem, 14
info_smm_mem_blocks, 18
init_smm_mem_info, 18
insert_block, 18
KB, 14
log_msg_error, 19
logical_to_physical_addr, 19
MB, 14
new_elem_list, 19
remove_all_blocks, 19
remove_block, 19
search_block, 20
search_max_hole, 20
set_log_start_addr, 20
set_next, 20
set_phy_start_addr, 21
set_pid, 21
set_size_block, 21
smmd_alloc, 21
smmd_expand, 22
smmd_realloc, 22
smmd_resize, 23
smmMem, 24
subset_allocated_blocks, 23
update_smm_mem_info, 23
write_to_log, 24
start
smm_blk, 8
subset_allocated_blocks
smmMem.h, 23
swap
smm_blk, 8

tokenizza
utility.h, 28

unlock_mutex
smm.h, 12
update_smm_mem_info
smmMem.h, 23
utility.h
command, 26, 27
FIFOFROM, 26
FIFOTO, 26
FILEPID, 26
FILEPIDSHELL, 26
fill_packet, 27
LINES, 26
MAX, 26
open_fifo, 27
open_fifos, 26
packet, 26
read_from, 27
reset_vett_cmd, 28
sm_alloc, 27
sm_calloc, 27
sm_check, 27
sm_exit, 27
sm_free, 27
sm_get, 27
sm_lab, 27
sm_lfb, 27
sm_malloc, 27
sm_not_found, 27
sm_put, 27
sm_realloc, 27
sm_start, 27
sm_stop, 27
tokenizza, 28
verifica_comando, 28
verifica_comando_server, 28
verifica_indirizzo_esadecimale, 29
verifica_valore, 29
write_to, 29

verifica_comando
utility.h, 28
verifica_comando_server
utility.h, 28
verifica_indirizzo_esadecimale
utility.h, 29
verifica_valore
utility.h, 29

write_to

utility.h, 29
write_to_log
smmMem.h, 24