OSMP - Entwurf und Implementierung einer Message Passing Umgebung für Interprozesskommunikation

Erstellt am 25.03.2024.

Erzeugt von Doxygen 1.10.0

Kapitel 1

Datei-Verzeichnis

1.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

Kapitel 2

Datei-Dokumentation

2.1 src/osmp_executables/ospExecutable_SendIRecv.c-Dateireferenz

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../osmp_library/OSMP.h"
```

Funktionen

- int main (int argc, char ∗argv[ ])

2.1.1 Dokumentation der Funktionen

2.1.1.1 main()

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

2.2 src/osmp_executables/ospExecutable_SendRecv.c-Dateireferenz

```
#include <stdio.h>
#include "../osmp_library/OSMP.h"
```

Funktionen

- int main (int argc, char ∗argv[ ])

2.2.1 Dokumentation der Funktionen

2.2.1.1 main()

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

2.3 src/osmp_library/OSMP.h-Dateireferenz

Makrodefinitionen

- #define OSMP_SUCCESS 0
- #define OSMP_FAILURE (!OSMP_SUCCESS)
- #define OSMP_MAX_MESSAGES_PROC 16
- #define OSMP_MAX_SLOTS 256
- #define OSMP_MAX_PAYLOAD_LENGTH 1024

Typdefinitionen

- typedef void * OSMP_Request
- typedef enum OSMP_Datatype OSMP_Datatype

Aufzählungen

- enum OSMP_Datatype {
  OSMP_SHORT , OSMP_INT , OSMP_LONG , OSMP_UNSIGNED_CHAR ,
  OSMP_UNSIGNED , OSMP_UNSIGNED_SHORT , OSMP_UNSIGNED_LONG , OSMP_FLOAT ,
  OSMP_DOUBLE , OSMP_BYTE }

Funktionen

- int get_OSMP_MAX_PAYLOAD_LENGTH ()
- int get_OSMP_MAX_SLOTS ()
- int get_OSMP_MAX_MESSAGES_PROC ()
- int get_OSMP_FAILURE ()
- int get_OSMP_SUCCESS ()
- int OSMP_SizeOf (OSMP_Datatype datatype, unsigned int *size)
- int OSMP_Init (const int *argc, char ***argv)
- int OSMP_Size (int *size)
- int OSMP_Rank (int *rank)
- int OSMP_Send (const void *buf, int count, OSMP_Datatype datatype, int dest)
- int OSMP_Recv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len)
- int OSMP_Finalize (void)
- int OSMP_Barrier (void)
- int OSMP_Gather (void *sendbuf, int sendcount, OSMP_Datatype sendtype, void *recvbuf, int recvcount, OSMP_Datatype recvtype, int recv)
- int OSMP_ISend (const void *buf, int count, OSMP_Datatype datatype, int dest, OSMP_Request request)
- int OSMP_IRecv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len, OSMP_Request request)
- int OSMP_Test (OSMP_Request request, int *flag)
- int OSMP_Wait (OSMP_Request request)
- int OSMP_CreateRequest (OSMP_Request *request)
- int OSMP_RemoveRequest (OSMP_Request *request)
- int OSMP_GetSharedMemoryName (char **name)

2.3.1 Makro-Dokumentation

2.3.1.1 OSMP_FAILURE

```
#define OSMP_FAILURE ( !OSMP_SUCCESS )
```

Im Fehlerfall liefern die OSMP-Funktionen den Wert OSMP_FAILURE zurück. Die Fehler führen aber nicht zum beenden des Programms (z. B. wenn ein Prozess eine Nachricht an einen nicht existierenden Prozess schickt).

2.3.1.2 OSMP_MAX_MESSAGES_PROC

```
#define OSMP_MAX_MESSAGES_PROC 16
```

Die maximale Zahl der Nachrichten pro Prozess

2.3.1.3 OSMP_MAX_PAYLOAD_LENGTH

```
#define OSMP_MAX_PAYLOAD_LENGTH 1024
```

Die maximale Länge der Nutzlast einer Nachricht

2.3.1.4 OSMP_MAX_SLOTS

```
#define OSMP_MAX_SLOTS 256
```

Die maximale Anzahl der Nachrichten, die insgesamt vorhanden sein dürfen

2.3.1.5 OSMP_SUCCESS

```
#define OSMP_SUCCESS 0
```

Alle OSMP-Funktionen liefern im Erfolgsfall OSMP_SUCCESS als Rückgabewert. Weitere Rückgabewerte können mit Begründung (und Dokumentation!) definiert werden

2.3.2 Dokumentation der benutzerdefinierten Typen

2.3.2.1 OSMP_Datatype

```
typedef enum OSMP_Datatype OSMP_Datatype
```

Die OSMP-Datentypen entsprechen den C-Datentypen. Sie werden verwendet, um den Typ der Daten anzugeben, die mit den OSMP-Funktionen gesendet bzw. empfangen werden sollen.

2.3.2.2 OSMP_Request

```
typedef void* OSMP_Request
```

2.3.3 Dokumentation der Aufzählungstypen

2.3.3.1 OSMP_Datatype

```
enum OSMP_Datatype
```

Die OSMP-Datentypen entsprechen den C-Datentypen. Sie werden verwendet, um den Typ der Daten anzugeben, die mit den OSMP-Funktionen gesendet bzw. empfangen werden sollen.

Aufzählungswerte

OSMP_SHORT	
OSMP_INT	
OSMP_LONG	
OSMP_UNSIGNED_CHAR	
OSMP_UNSIGNED	
OSMP_UNSIGNED_SHORT	
OSMP_UNSIGNED_LONG	
OSMP_FLOAT	
OSMP_DOUBLE	
OSMP_BYTE	

2.3.4 Dokumentation der Funktionen

2.3.4.1 get_OSMP_FAILURE()

```
int get_OSMP_FAILURE ( )
```

Gibt den Wert von OSMP_FAILURE zurück.

2.3.4.2 get_OSMP_MAX_MESSAGES_PROC()

```
int get_OSMP_MAX_MESSAGES_PROC ( )
```

Gibt die maximale Zahl der Nachrichten pro Prozess zurück.

2.3.4.3 get_OSMP_MAX_PAYLOAD_LENGTH()

```
int get_OSMP_MAX_PAYLOAD_LENGTH ( )
```

Gibt die maximale Länge der Nutzlast einer Nachricht zurück.

In dieser Quelltext-Datei sind Implementierungen der OSMP Bibliothek zu finden.

2.3.4.4 get_OSMP_MAX_SLOTS()

```
int get_OSMP_MAX_SLOTS ( )
```

Gibt die Maximale Anzahl der Nachrichten, die insgesamt vorhanden sein dürfen zurück.

2.3.4.5 get_OSMP_SUCCESS()

```
int get_OSMP_SUCCESS ( )
```

Gibt den Wert von OSMP_SUCCESS zurück.

2.3.4.6 OSMP_Barrier()

```
int OSMP_Barrier (
            void  )
```

Diese kollektive Funktion blockiert den aufrufenden Prozess. Erst wenn alle anderen Prozesse ebenfalls an der Barriere angekommen sind, laufen die Prozesse weiter.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.7 OSMP_CreateRequest()

```
int OSMP_CreateRequest (
            OSMP_Request * request )
```

Erstellt eine OSMP_Request. Eine OSMP_Request wird dazu verwendet, um nicht blockierende Operationen zu überwachen.

Parameter

out	*request*	Adresse eines Requests (input)

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.8 OSMP_Finalize()

```
int OSMP_Finalize (
            void  )
```

Alle OSMP-Prozesse müssen diese Funktion aufrufen, bevor sie sich beenden. Sie geben damit den Zugriff auf die gemeinsamen Ressourcen frei. Hierbei muss jeder Prozess zuvor alle noch vorhandenen Nachrichten abarbeiten. Dies bedeutet, dass der Posteingang gesperrt wird und alle noch vorhandenen Nachrichten gelöscht werden.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.9 OSMP_Gather()

```
int OSMP_Gather (
            void * sendbuf,
            int sendcount,
            OSMP_Datatype sendtype,
            void * recvbuf,
            int recvcount,
            OSMP_Datatype recvtype,
            int recv )
```

Diese Funktion ermöglicht die Gather-Kommunikation. Hierbei können mehrere Prozesse an einen Empfänger Prozess Daten schicken.

Parameter

in	*sendbuf*	Zeiger auf den Sendepuffer.
in	*sendcount*	Anzahl der Elemente im Sendepuffer.
in	*sendtype*	OSMP-Datentyp der Elemente im Sendepuffer.
out	*recvbuf*	Zeiger auf den Empfangspuffer.
in	*recvcount*	Anzahl der Elemente im Empfangspuffer.
in	*recvtype*	OSMP-Datentyp der Elemente im Empfangspuffer.
in	*recv*	1, falls der aufrufende Prozess der Empfänger ist, sonst 0.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.10 OSMP_GetSharedMemoryName()

```
int OSMP_GetSharedMemoryName (
            char ** name )
```

Gibt den Namen des Shared Memory Bereichs zurück.

Parameter

out	*name*	Der Name des Shared Memory Bereichs

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.11 OSMP_Init()

```
int OSMP_Init (
            const int * argc,
            char *** argv )
```

Die Funktion OSMP_Init() initialisiert die OSMP-Umgebung und ermöglicht den Zugang zu den gemeinsamen Ressourcen der OSMP-Prozesse. Sie muss von jedem OSMP-Prozess zu Beginn aufgerufen werden. Durch diesen Aufruf wird außerdem der Posteingang des Prozesses freigegeben.

Parameter

in	*argc*	Adresse der Argumentzahl
in	*argv*	Adresse des Argumentvektors

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.12 OSMP_IRecv()

```
int OSMP_IRecv (
            void * buf,
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len,
            OSMP_Request request )
```

Die Funktion empfängt eine Nachricht analog zu OSMP_Recv(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Empfangen).

Parameter

out	*buf*	Startadresse des Speicherbereichs, wo die zu empfangende Nachricht gespeichert werden soll.
in	*count*	Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	*datatype*	OSMP-Typ der Daten im Puffer
out	*source*	PID des Senders zwischen 0, . . . , np-1
out	*len*	tatsächliche Länge der empfangenen Nachricht in Byte
in,out	*request*	Adresse einer Datenstruktur, die später verwendet werden kann, um abzufragen, ob die die Operation abgeschlossen ist.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.13 OSMP_ISend()

```
int OSMP_ISend (
            const void * buf,
            int count,
            OSMP_Datatype datatype,
            int dest,
            OSMP_Request request )
```

Die Funktion sendet eine Nachricht analog zu OSMP_Send(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Senden).

Parameter

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	PID des Empfängers zwischen 0, . . . , np-1
in,out	*request*	Adresse einer eigenen Datenstruktur, die später verwendet werden kann, um abzufragen, ob die Operation abgeschlossen ist.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.14 OSMP_Rank()

```
int OSMP_Rank (
            int * rank )
```

Die Funktion OSMP_Rank() liefert in ∗rank die OSMP-Prozessnummer des aufrufenden OSMP-Prozesses von 0,. . . ,np-1 zurück.

Parameter

out	*rank*	Prozessnummer 0,. . . ,np-1 des aktuellen OSMP-Prozesse

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.15 OSMP_Recv()

```
int OSMP_Recv (
            void * buf,
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len )
```

Der aufrufende Prozess empfängt eine Nachricht mit maximal count Elementen des angegebenen Datentyps datatype. Die Nachricht wird an die Adresse buf des aufrufenden Prozesses geschrieben. Unter source wird die OSMP-Prozessnummer des sendenden Prozesses und unter len die tatsächliche Länge der gelesenen Nachricht abgelegt. Die Funktion ist blockierend, d.h. sie wartet, bis eine Nachricht für den Prozess vorhanden ist. Wenn die Funktion zurückkehrt, ist der Kopierprozess abgeschlossen. Die Nachricht gilt nach dem Aufruf dieser Funktion als abgearbeitet.

Parameter

out	*buf*	Startadresse des Puffers im lokalen Speicher des aufrufenden Prozesses, in den die Nachricht kopiert werden soll.
in	*count*	maximale Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	*datatype*	OSMP-Typ der Daten im Puffer
out	*source*	Nummer des Senders zwischen 0,. . . ,np-1
out	*len*	tatsächliche Länge der empfangenen Nachricht in Byte

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.16 OSMP_RemoveRequest()

```
int OSMP_RemoveRequest (
            OSMP_Request * request )
```

Löscht eine OSMP_Request.

Parameter

in	*request*	Adresse eines Requests

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.17 OSMP_Send()

```
int OSMP_Send (
            const void * buf,
            int count,
            OSMP_Datatype datatype,
            int dest )
```

Die Funktion OSMP_Send() sendet eine Nachricht an den Prozess mit der Nummer dest. Die Nachricht besteht aus count Elementen vom Typ datatype. Die zu sendende Nachricht beginnt im aufrufenden Prozess bei der Adresse buf. Die Funktion ist blockierend, d.h. wenn sie in das aufrufende Programm zurückkehrt, ist der Kopiervorgang abgeschlossen.

Parameter

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	Nummer des Empfängers zwischen 0,. . . ,np-1

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.18 OSMP_Size()

```
int OSMP_Size (
            int * size )
```

Die Funktion OSMP_Size() liefert in *size* die Zahl der OSMP-Prozesse ohne den OSMP-Starter Prozess zurück. Sollte mit der Zahl übereinstimmen, die in der Kommandozeile dem OSMP-Starter übergeben wird.

Parameter

out	*size*	Zahl der OSMP-Prozesse

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.19 OSMP_SizeOf()

```
int OSMP_SizeOf (
            OSMP_Datatype datatype,
            unsigned int * size )
```

Die Funktion OSMP_SizeOf() liefert in *size* die Größe des Datentyps datatype in Byte zurück.

Parameter

in	*datatype*	OSMP-Datentyp
out	*size*	Größe des Datentyps in Byte

Rückgabe

Im Erfolgsfall OSMP_SUCCESS; falls der OSMP_Datatype nicht existiert, OSMP_FAILURE

2.3.4.20 OSMP_Test()

```
int OSMP_Test (
            OSMP_Request request,
            int * flag )
```

Die Funktion testet, ob die mit der Request verknüpften Operation abgeschlossen ist. Sie ist nicht blockierend, d.h. sie wartet nicht auf das Ende der mit request verknüpften Operation.

Parameter

in	*request*	Adresse der Struktur, die eine blockierende Operation spezifiziert
out	*flag*	Gibt den Status der Operation an.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.21 OSMP_Wait()

```
int OSMP_Wait (
            OSMP_Request request )
```

Die Funktion wartet, bis die mit der Request verknüpfte, nicht blockierende Operation abgeschlossen ist. Sie ist so lange blockiert, bis dies der Fall ist.

Parameter

in	*request*	Adresse der Struktur, die eine nicht blockierende Operation spezifiziert

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.4 OSMP.h

gehe zur Dokumentation dieser Datei

```
00001 /*****************************************************************************
00002 * FILE: OSMP.h
00003 * DESCRIPTION:
00004 * Beinhaltet alle in der Anleitung angegeben Prototypen der OSMP Kernfunktionen wie z.B. OSMP_Test()
      und vorgegebene Konstanten.
00005 *
00006 * LAST MODIFICATION: March 07, 2024
00007 *****************************************************************************/
00008 #ifndef BETRIEBSSYSTEME_OSMP_H
00009 #define BETRIEBSSYSTEME_OSMP_H
00010
00015 #define OSMP_SUCCESS 0
00016
00021 #define OSMP_FAILURE ( !OSMP_SUCCESS )
00022
00023 typedef void* OSMP_Request;
00024
00028 #define OSMP_MAX_MESSAGES_PROC 16
00029
00033 #define OSMP_MAX_SLOTS 256
00034
00038 #define OSMP_MAX_PAYLOAD_LENGTH 1024
00039
00044 typedef enum OSMP_Datatype {
00045     OSMP_SHORT,          // short int
00046     OSMP_INT,            // int
00047     OSMP_LONG,           // long int
00048     OSMP_UNSIGNED_CHAR,  // unsigned char
00049     OSMP_UNSIGNED,       // unsigned
00050     OSMP_UNSIGNED_SHORT,// unsigned short int
00051     OSMP_UNSIGNED_LONG,  // unsigned long int
00052     OSMP_FLOAT,          // float
00053     OSMP_DOUBLE,         // double
00054     OSMP_BYTE            // char
00055 } OSMP_Datatype;
00056
00060 int get_OSMP_MAX_PAYLOAD_LENGTH();
00061
00065 int get_OSMP_MAX_SLOTS();
00066
00070 int get_OSMP_MAX_MESSAGES_PROC();
00071
00075 int get_OSMP_FAILURE();
00076
00080 int get_OSMP_SUCCESS();
00081
00090 int OSMP_SizeOf(OSMP_Datatype datatype, unsigned int *size);
00091
00102 int OSMP_Init(const int *argc, char ***argv);
00103
00112 int OSMP_Size(int *size);
00113
00121 int OSMP_Rank(int *rank);
00122
00136 int OSMP_Send(const void *buf, int count, OSMP_Datatype datatype, int dest);
00137
00154 int OSMP_Recv(void *buf, int count, OSMP_Datatype datatype, int *source, int *len);
00155
00163 int OSMP_Finalize(void);
00164
00171 int OSMP_Barrier(void);
```

```
00172
00187 int OSMP_Gather(void *sendbuf, int sendcount, OSMP_Datatype sendtype, void *recvbuf, int recvcount,
      OSMP_Datatype recvtype, int recv);
00188
00201 int OSMP_ISend(const void *buf, int count, OSMP_Datatype datatype, int dest, OSMP_Request request);
00202
00216 int OSMP_IRecv(void *buf, int count, OSMP_Datatype datatype, int *source, int *len, OSMP_Request
      request);
00217
00227 int OSMP_Test(OSMP_Request request, int *flag);
00228
00237 int OSMP_Wait(OSMP_Request request);
00238
00247 int OSMP_CreateRequest(OSMP_Request *request);
00248
00256 int OSMP_RemoveRequest(OSMP_Request *request);
00257
00265 int OSMP_GetSharedMemoryName(char **name);
00266
00267 #endif /* BETRIEBSSYSTEME_OSMP_H */
```

2.5 src/osmp_library/osmplib.c-Dateireferenz

```
#include "osmplib.h"
```

Funktionen

- int get_OSMP_MAX_PAYLOAD_LENGTH ()
- int get_OSMP_MAX_SLOTS ()
- int get_OSMP_MAX_MESSAGES_PROC ()
- int get_OSMP_FAILURE ()
- int get_OSMP_SUCCESS ()
- int OSMP_Init (const int *argc, char ***argv)
- int OSMP_SizeOf (OSMP_Datatype datatype, unsigned int *size)
- int OSMP_Size (int *size)
- int OSMP_Rank (int *rank)
- int OSMP_Send (const void *buf, int count, OSMP_Datatype datatype, int dest)
- int OSMP_Recv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len)
- int OSMP_Finalize (void)
- int OSMP_Barrier (void)
- int OSMP_Gather (void *sendbuf, int sendcount, OSMP_Datatype sendtype, void *recvbuf, int recvcount, OSMP_Datatype recvtype, int recv)
- int OSMP_ISend (const void *buf, int count, OSMP_Datatype datatype, int dest, OSMP_Request request)
- int OSMP_IRecv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len, OSMP_Request request)
- int OSMP_Test (OSMP_Request request, int *flag)
- int OSMP_Wait (OSMP_Request request)
- int OSMP_CreateRequest (OSMP_Request *request)
- int OSMP_RemoveRequest (OSMP_Request *request)
- int OSMP_GetSharedMemoryName (char **name)

2.5.1 Dokumentation der Funktionen

2.5.1.1 get_OSMP_FAILURE()

```
int get_OSMP_FAILURE ( )
```

Gibt den Wert von OSMP_FAILURE zurück.

2.5.1.2 get_OSMP_MAX_MESSAGES_PROC()

```
int get_OSMP_MAX_MESSAGES_PROC ( )
```

Gibt die maximale Zahl der Nachrichten pro Prozess zurück.

2.5.1.3 get_OSMP_MAX_PAYLOAD_LENGTH()

```
int get_OSMP_MAX_PAYLOAD_LENGTH ( )
```

In dieser Quelltext-Datei sind Implementierungen der OSMP Bibliothek zu finden.

2.5.1.4 get_OSMP_MAX_SLOTS()

```
int get_OSMP_MAX_SLOTS ( )
```

Gibt die Maximale Anzahl der Nachrichten, die insgesamt vorhanden sein dürfen zurück.

2.5.1.5 get_OSMP_SUCCESS()

```
int get_OSMP_SUCCESS ( )
```

Gibt den Wert von OSMP_SUCCESS zurück.

2.5.1.6 OSMP_Barrier()

```
int OSMP_Barrier (
            void  )
```

Diese kollektive Funktion blockiert den aufrufenden Prozess. Erst wenn alle anderen Prozesse ebenfalls an der Barriere angekommen sind, laufen die Prozesse weiter.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.7 OSMP_CreateRequest()

```
int OSMP_CreateRequest (
            OSMP_Request * request )
```

Erstellt eine OSMP_Request. Eine OSMP_Request wird dazu verwendet, um nicht blockierende Operationen zu überwachen.

Parameter

out	*request*	Adresse eines Requests (input)

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.8 OSMP_Finalize()

```
int OSMP_Finalize (
            void  )
```

Alle OSMP-Prozesse müssen diese Funktion aufrufen, bevor sie sich beenden. Sie geben damit den Zugriff auf die gemeinsamen Ressourcen frei. Hierbei muss jeder Prozess zuvor alle noch vorhandenen Nachrichten abarbeiten. Dies bedeutet, dass der Posteingang gesperrt wird und alle noch vorhandenen Nachrichten gelöscht werden.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.9 OSMP_Gather()

```
int OSMP_Gather (
            void * sendbuf,
            int sendcount,
            OSMP_Datatype sendtype,
            void * recvbuf,
            int recvcount,
            OSMP_Datatype recvtype,
            int recv )
```

Diese Funktion ermöglicht die Gather-Kommunikation. Hierbei können mehrere Prozesse an einen Empfänger Prozess Daten schicken.

Parameter

in	*sendbuf*	Zeiger auf den Sendepuffer.
in	*sendcount*	Anzahl der Elemente im Sendepuffer.
in	*sendtype*	OSMP-Datentyp der Elemente im Sendepuffer.
out	*recvbuf*	Zeiger auf den Empfangspuffer.
in	*recvcount*	Anzahl der Elemente im Empfangspuffer.
in	*recvtype*	OSMP-Datentyp der Elemente im Empfangspuffer.
in	*recv*	1, falls der aufrufende Prozess der Empfänger ist, sonst 0.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.10 OSMP_GetSharedMemoryName()

```
int OSMP_GetSharedMemoryName (
            char ** name )
```

Gibt den Namen des Shared Memory Bereichs zurück.

Parameter

out	*name*	Der Name des Shared Memory Bereichs

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.11 OSMP_Init()

```
int OSMP_Init (
            const int * argc,
            char *** argv )
```

Die Funktion OSMP_Init() initialisiert die OSMP-Umgebung und ermöglicht den Zugang zu den gemeinsamen Ressourcen der OSMP-Prozesse. Sie muss von jedem OSMP-Prozess zu Beginn aufgerufen werden. Durch diesen Aufruf wird außerdem der Posteingang des Prozesses freigegeben.

Parameter

in	*argc*	Adresse der Argumentzahl
in	*argv*	Adresse des Argumentvektors

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.12 OSMP_IRecv()

```
int OSMP_IRecv (
            void * buf,
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len,
            OSMP_Request request )
```

Die Funktion empfängt eine Nachricht analog zu OSMP_Recv(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Empfangen).

Parameter

out	*buf*	Startadresse des Speicherbereichs, wo die zu empfangende Nachricht gespeichert werden soll.
in	*count*	Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	*datatype*	OSMP-Typ der Daten im Puffer
out	*source*	PID des Senders zwischen 0, . . . , np-1
out	*len*	tatsächliche Länge der empfangenen Nachricht in Byte
in,out	*request*	Adresse einer Datenstruktur, die später verwendet werden kann, um abzufragen, ob die die Operation abgeschlossen ist.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.13 OSMP_ISend()

```
int OSMP_ISend (
            const void * buf,
            int count,
            OSMP_Datatype datatype,
            int dest,
            OSMP_Request request )
```

Die Funktion sendet eine Nachricht analog zu OSMP_Send(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Senden).

Parameter

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	PID des Empfängers zwischen 0, . . . , np-1
in,out	*request*	Adresse einer eigenen Datenstruktur, die später verwendet werden kann, um abzufragen, ob die Operation abgeschlossen ist.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.14 OSMP_Rank()

```
int OSMP_Rank (
            int * rank )
```

Die Funktion OSMP_Rank() liefert in ∗rank die OSMP-Prozessnummer des aufrufenden OSMP-Prozesses von 0,. . . ,np-1 zurück.

Parameter

out	*rank*	Prozessnummer 0,. . . ,np-1 des aktuellen OSMP-Prozesse

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.15 OSMP_Recv()

```
int OSMP_Recv (
            void * buf,
```

```
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len )
```

Der aufrufende Prozess empfängt eine Nachricht mit maximal count Elementen des angegebenen Datentyps datatype. Die Nachricht wird an die Adresse buf des aufrufenden Prozesses geschrieben. Unter source wird die OSMP-Prozessnummer des sendenden Prozesses und unter len die tatsächliche Länge der gelesenen Nachricht abgelegt. Die Funktion ist blockierend, d.h. sie wartet, bis eine Nachricht für den Prozess vorhanden ist. Wenn die Funktion zurückkehrt, ist der Kopierprozess abgeschlossen. Die Nachricht gilt nach dem Aufruf dieser Funktion als abgearbeitet.

Parameter

out	*buf*	Startadresse des Puffers im lokalen Speicher des aufrufenden Prozesses, in den die Nachricht kopiert werden soll.
in	*count*	maximale Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	*datatype*	OSMP-Typ der Daten im Puffer
out	*source*	Nummer des Senders zwischen 0,. . . ,np-1
out	*len*	tatsächliche Länge der empfangenen Nachricht in Byte

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.16 OSMP_RemoveRequest()

```
int OSMP_RemoveRequest (
            OSMP_Request * request )
```

Löscht eine OSMP_Request.

Parameter

in	*request*	Adresse eines Requests

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.17 OSMP_Send()

```
int OSMP_Send (
            const void * buf,
            int count,
            OSMP_Datatype datatype,
            int dest )
```

Die Funktion OSMP_Send() sendet eine Nachricht an den Prozess mit der Nummer dest. Die Nachricht besteht aus count Elementen vom Typ datatype. Die zu sendende Nachricht beginnt im aufrufenden Prozess bei der Adresse buf. Die Funktion ist blockierend, d.h. wenn sie in das aufrufende Programm zurückkehrt, ist der Kopiervorgang abgeschlossen.

Parameter

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	Nummer des Empfängers zwischen 0,. . . ,np-1

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.18 OSMP_Size()

```
int OSMP_Size (
            int * size )
```

Die Funktion OSMP_Size() liefert in *size* die Zahl der OSMP-Prozesse ohne den OSMP-Starter Prozess zurück. Sollte mit der Zahl übereinstimmen, die in der Kommandozeile dem OSMP-Starter übergeben wird.

Parameter

out	*size*	Zahl der OSMP-Prozesse

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.19 OSMP_SizeOf()

```
int OSMP_SizeOf (
            OSMP_Datatype datatype,
            unsigned int * size )
```

Die Funktion OSMP_SizeOf() liefert in *size* die Größe des Datentyps datatype in Byte zurück.

Parameter

in	*datatype*	OSMP-Datentyp
out	*size*	Größe des Datentyps in Byte

Rückgabe

Im Erfolgsfall OSMP_SUCCESS; falls der OSMP_Datatype nicht existiert, OSMP_FAILURE

2.5.1.20 OSMP_Test()

```
int OSMP_Test (
```

```
            OSMP_Request request,
            int * flag )
```

Die Funktion testet, ob die mit der Request verknüpften Operation abgeschlossen ist. Sie ist nicht blockierend, d.h. sie wartet nicht auf das Ende der mit request verknüpften Operation.

Parameter

in	*request*	Adresse der Struktur, die eine blockierende Operation spezifiziert
out	*flag*	Gibt den Status der Operation an.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.5.1.21 OSMP_Wait()

```
int OSMP_Wait (
            OSMP_Request request )
```

Die Funktion wartet, bis die mit der Request verknüpfte, nicht blockierende Operation abgeschlossen ist. Sie ist so lange blockiert, bis dies der Fall ist.

Parameter

in	*request*	Adresse der Struktur, die eine nicht blockierende Operation spezifiziert

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.6 src/osmp_library/osmplib.h-Dateireferenz

```
#include "OSMP.h"
```

Makrodefinitionen

- #define UNUSED(x) { (void)(x); }

2.6.1 Makro-Dokumentation

2.6.1.1 UNUSED

```
#define UNUSED(
             x ) { (void)(x); }
```

Dieses Makro wird verwendet, um den Compiler davon zu überzeugen, dass eine Variable verwendet wird.

2.7 osmplib.h

gehe zur Dokumentation dieser Datei

```
00001 #ifndef BETRIEBSSYSTEME_OSMPLIB_H
00002 #define BETRIEBSSYSTEME_OSMPLIB_H
00003
00004 #include "OSMP.h"
00005
00009 #define UNUSED(x) { (void)(x); }
00010
00011 #endif //BETRIEBSSYSTEME_OSMPLIB_H
```

2.8 src/osmp_runner/osmp_run.c-Dateireferenz

```
#include "osmp_run.h"
```

Funktionen

- int main (int argc, char **argv)

2.8.1 Dokumentation der Funktionen

2.8.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

2.9 src/osmp_runner/osmp_run.h-Dateireferenz

```
#include <stdio.h>
#include "../osmp_library/OSMP.h"
```

2.10 osmp_run.h

gehe zur Dokumentation dieser Datei

```
00001 #ifndef OSMP_RUN_H
00002 #define OSMP_RUN_H
00003
00004 #include <stdio.h>
00005
00006 #include "../osmp_library/OSMP.h"
00007
00008 #endif // OSMP_RUN_H
```

Index