Коммуникационные операции «точка-точка»

параллельное программирование

Коммуникационные операции типа «точка-точка»

К операциям этого типа относятся две представленные выше коммуникационные процедуры (MPI_Send, MPI_Recv). В коммуникационных операциях типа точка-точка всегда участвуют 2 процесса: передающий и принимающий. В MPI имеется множество функций, реализующих такой тип обмена. Многообразие объясняется возможностью организации таких обменов множеством способов. Описанные в предыдущем разделе функции реализуют *стандартный режим с блокировкой.*

Блокирующие функции

- *Блокирующие функции* подразумевают полное окончание операции после выхода из процедуры, т.е. вызывающий процесс блокируется, пока операция не будет завершена.

Неблокирующие функции

- *Неблокирующие функции* подразумевают совмещение операций обмена с другими операциями, поэтому неблокирующие функции передачи и приема по сути дела являются функциями инициализации соответствующих операций. Для опроса завершенности операции (и принудительного завершения) вводятся дополнительные функции.

Коммуникационные операции типа «точка-точка»

Способ связи	С блокировкой	Без блокировки
Стандартная посылка	`MPI_Send`	`MPI_Isend`
Синхронная посылка	`MPI_Ssend`	`MPI_Issend`
Буферизированная посылка	`MPI_Bsend`	`MPI_Ibsend`
Согласованная посылка	`MPI_Rsend`	`MPI_Irsend`
Прием информации	`MPI_Recv`	`MPI_Irecv`

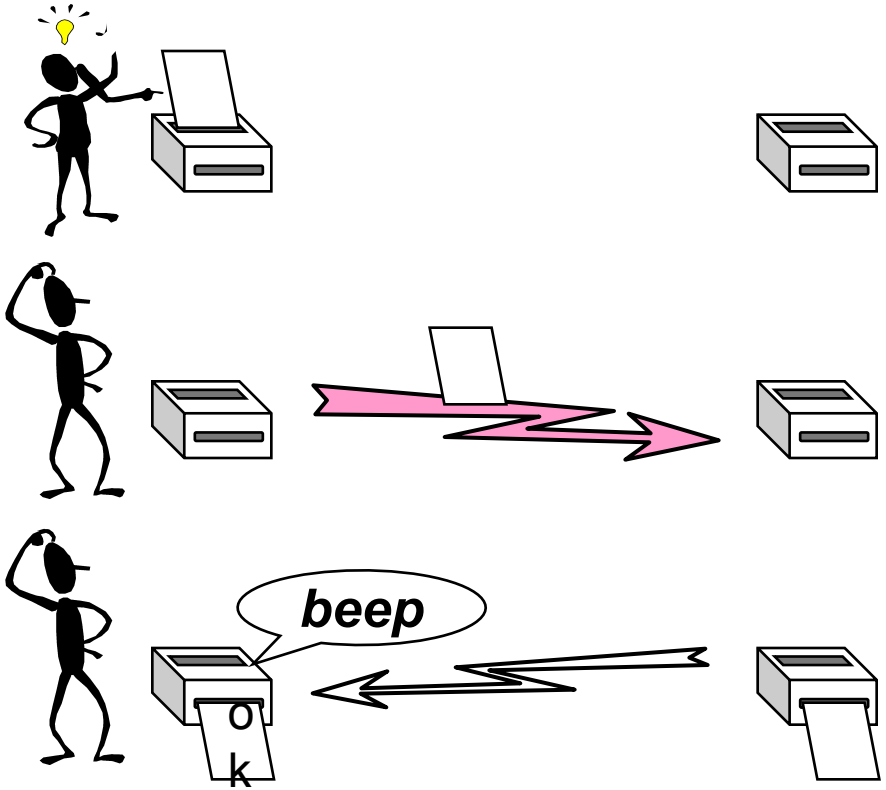
- Из таблицы хорошо виден принцип формирования имен функций. К именам базовых функций Send/Recv добавляются различные префиксы.

Коммуникационные операции типа «точка-точка»

Префикс S	(synchronous) – означает синхронный режим передачи данных. Операция передачи данных заканчивается только тогда, когда заканчивается прием данных.
Префикс B	(buffered) – означает буферизованный режим передачи данных. В адресном пространстве передающего процесса с помощью специальной функции (MPI_Buffer_attach) создается буфер обмена, который используется в операциях обмена. Операция посылки заканчивается, когда данные помещены в этот буфер.
Префикс R	(ready) – согласованный или подготовленный режим передачи данных. Операция передачи данных должна начинаться только тогда, когда принимающий процессор **уже точно** выставил признак готовности приема данных.
Префикс I	(immediate) – относится к неблокирующим операциям.

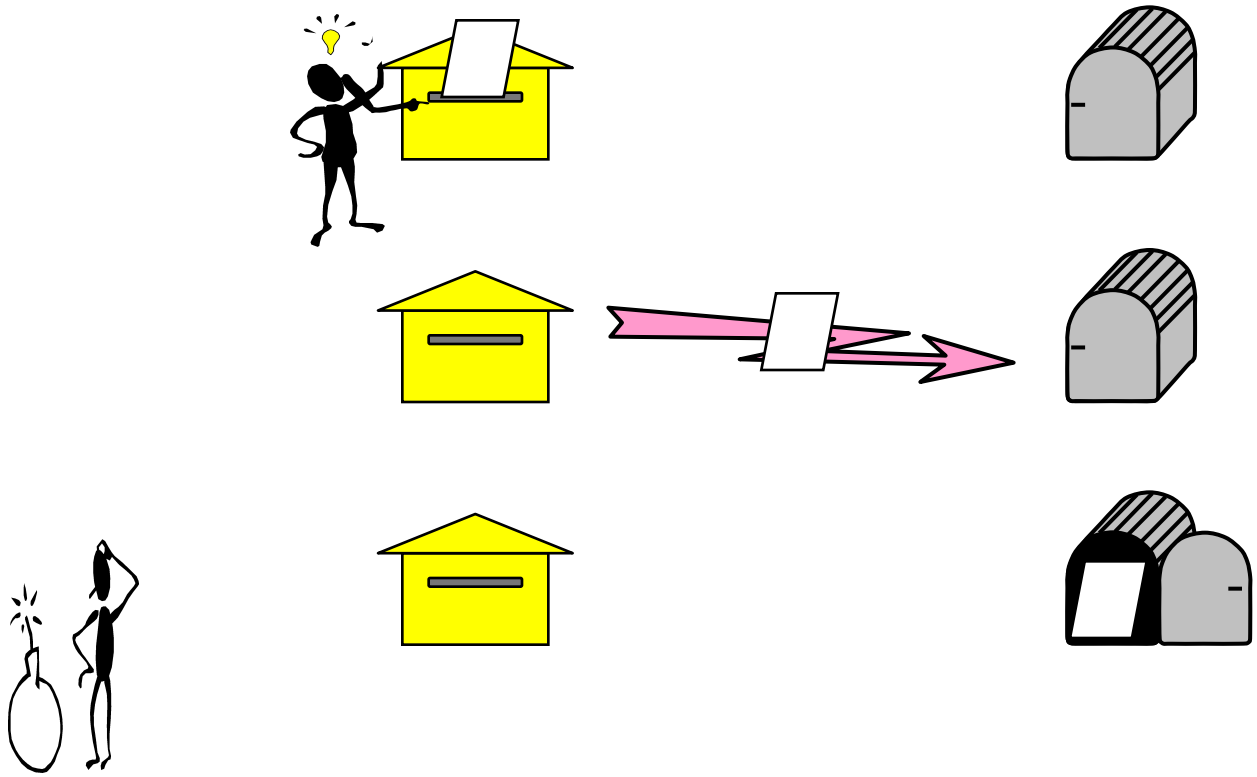
MPI – Синхронная посылка

- Процессор-отправитель ожидает информацию о том, когда получатель примет сообщение.
- Пример, факс получатель присылает тег завершения приема.



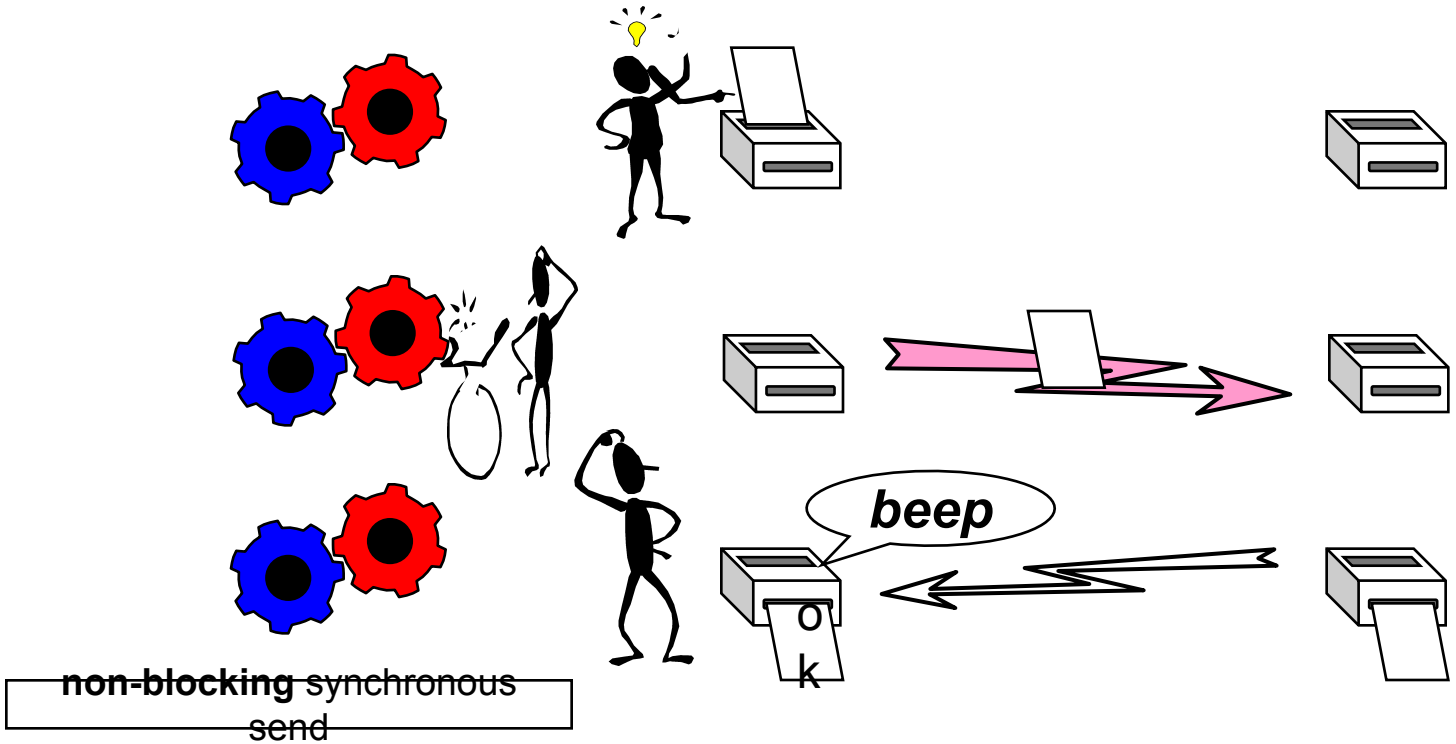
MPI – Буферизированная посылка или Несинхронная посылка

- Процессор-отправитель знает только когда сообщение ушло.



MPI–посылки без блокировки

- Неблокирующие операции немедленно возвращают управление программе. Программа выполняет следующие действия.
- Для того, что бы спустя некоторое время убедиться, что неблокирующая функция передачи данных выполнена полностью, нужно вызвать функцию MPI_Test или MPI_Wait.



Неблокирующие коммуникационные операции

- Использование неблокирующих коммуникационных операций более безопасно с точки зрения возникновения тупиковых ситуаций, а также может увеличить скорость работы программы за счет совмещения выполнения вычислительных и коммуникационных операций. Эти задачи решаются разделением коммуникационных операций на две стадии: инициирование операции и проверку завершения операции.

Неблокирующие коммуникационные операции

- Неблокирующие операции используют специальный скрытый (opaque) объект "запрос обмена" (request) для связи между функциями обмена и функциями опроса их завершения.
- Для прикладных программ доступ к этому объекту возможен только через вызовы MPI-функций.
- Если операция обмена завершена, подпрограмма проверки снимает "запрос обмена", устанавливая его в значение MPI_REQUEST_NULL.
- Снять "запрос обмена" без ожидания завершения операции можно подпрограммой MPI_Request_free.

Функция передачи сообщений **MPI_Isend**

```
int MPI_Isend( void *sbuf, int count, MPI_Datatype
datatype, int dest, int tag, MPI_Comm comm,
MPI_Request *request )
```

- Входные параметры:

sbuf	адрес начала расположения пересылаемых данных
count	число пересылаемых элементов
datatype	тип посылаемых элементов
dest	номер процесса-получателя в группе, связанной с коммуникатором comm
tag	идентификатор сообщения
comm	коммуникатор области связи

- Выходные параметры:

request	запрос обмена

Функция приема сообщений **MPI_Irecv**

```
int MPI_Irecv( void *rbuf, int count, MPI_Datatype
datatype, int source, int tag, MPI_Comm comm,
MPI_Request *request )
```

- Входные параметры:

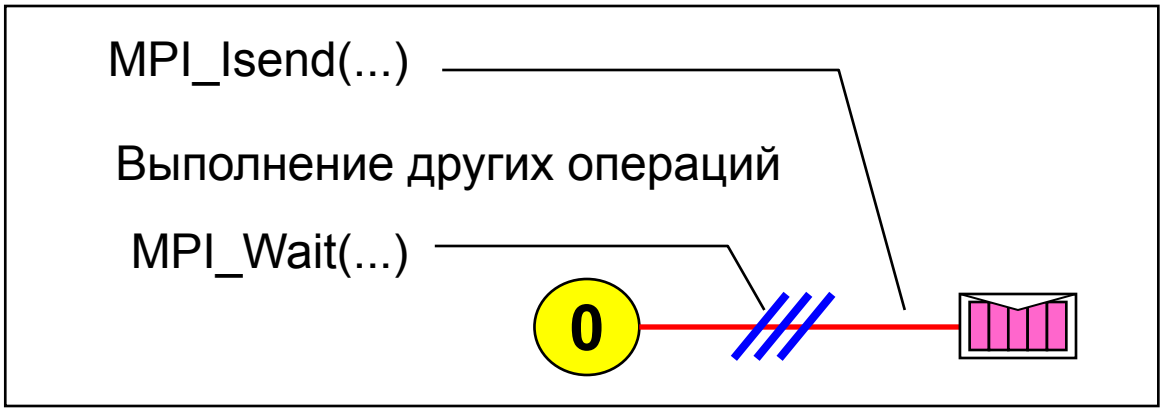
`count`	число пересылаемых элементов
`datatype`	тип посылаемых элементов
`source`	номер процесса-отправителя
`tag`	идентификатор сообщения
`comm`	коммуникатор области связи

- Выходные параметры:

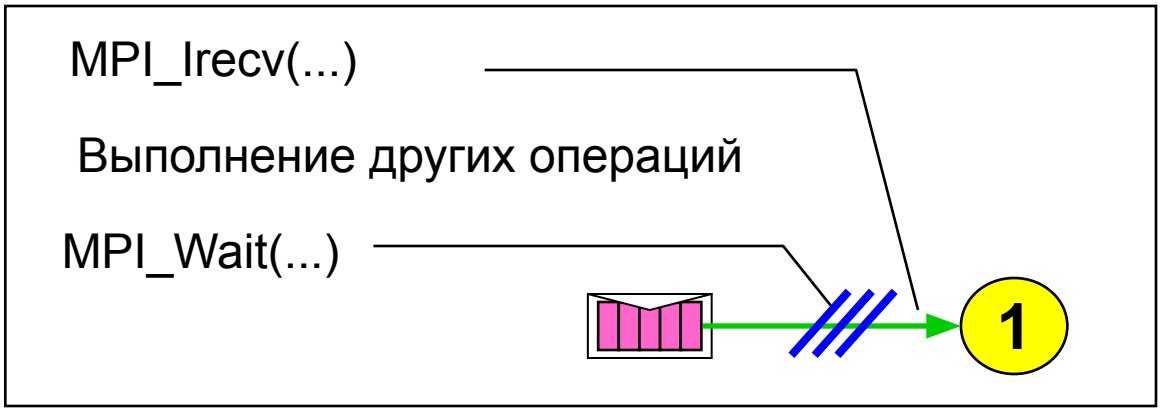
`rbuf`	адрес начала расположения принимаемого сообщения
`request`	запрос обмена

MPI – Non-Blocking Examples

- неблокирующая посылка



- неблокирующий прием



/// = прекращается выполнение операций пока не завершится передача/прием

Функция ожидания завершения неблокирующей операции **MPI_Wait**

```
int MPI_Wait ( MPI_Request  *request, MPI_Status
*status)
```

- Входные параметры:

`request`	запрос обмена

- Выходные параметры:

`status`	атрибуты принятого сообщений

Функция проверки завершения неблокирующей операции **MPI_Test**

```
int MPI_Test ( MPI_Request  *request, int *flag,
MPI_Status *status)
```

- Входные параметры:

`request`	запрос обмена

- Выходные параметры:

`status`	атрибуты принятого сообщений
`flag`	признак завершенности проверяемой операции *flag*==0, если операция не завершена *flag*<>0, если операция завершена

MPI_Sendrecv – совмещение отправки и получения

```
int MPI_Sendrecv(void *sendbuf, int sendcount,
MPI_Datatype sendtype, int dest, int sendtag, void
*recvbuf, int recvcount, MPI_Datatype recvtype, int
source, int recvtag, MPI_Comm comm, MPI_Status *status)
```

- Выходные параметры:

recvbuf	адрес начала расположения принимаемого сообщения
status	атрибуты принятого сообщения

- При использовании блокирующего режима передачи сообщений существует потенциальная опасность возникновения тупиковых ситуаций, в которых операции обмена данными блокируют друг друга.
- В ситуациях, когда требуется выполнить *взаимный обмен данными между процессами*, безопаснее использовать *совмещенную операцию MPI_Sendrecv*.

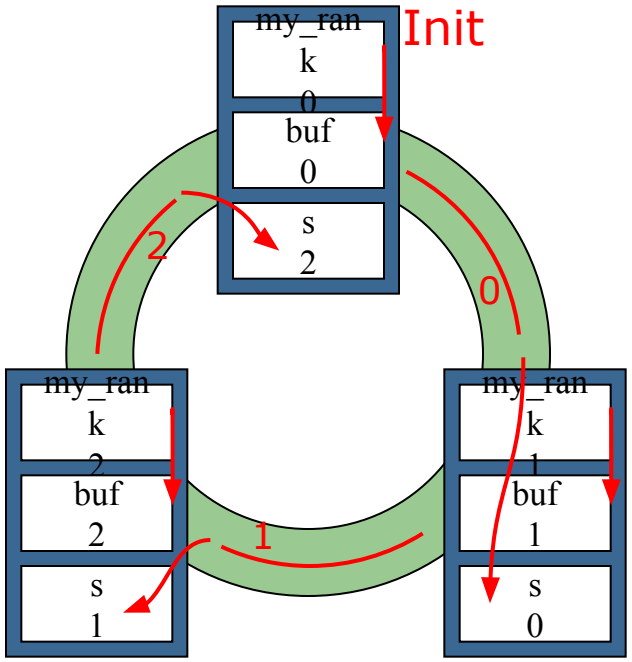
Задание1

- Нулевой процесс выполняет продолжительный цикл и после его выполнения посылает первому процессору значение вычислений цикла при помощи коммуникационной функции MPI_Send;
- A) Первый процесс засекает время t1, выполняет блокирующую функцию MPI_**Recv**, засекает время t2, выводит присланное значение и затраченное время на ожидание и прием посылки.
- B) Первый процесс засекает время t1, выполняет неблокирующую функцию **MPI_Irecv**, засекает время t2, выводит полученное значение и затраченное время на выполнение неблокирующей операции приема.
- C) Первый процессор засекает время t1, выполняет неблокирующую функцию **MPI_Irecv**, засекает время t2, выполняет операцию **MPI_Wait**, засекает время t3, выводит полученное значение и затраченное время на выполнение неблокирующей операции приема и ожидание получения посылки.

Задание2 — Пересылка данных по кольцу

- Каждый процессор помещает свой ранг в целочисленную переменную *buf*.
- Каждый процессор пересылает переменную *buf* соседу справа.
- Каждый процессор суммирует принимаемое значение в переменную *s*, а затем передаёт принятое значение соседу справа.
- Пересылки по кольцу прекращаются, когда каждый процессор получит то значение, с которого начал пересылки:
  - т.е. каждый процессор просуммирует ранги всех процессоров.
- С целью исключения взаимоблокировки используются неблокирующие пересылки **MPI_Isend**.

```
next = (my_id+1) % np;
prev = (my_id-1+np) % np;
```



Задание 3 — Пересылка данных по кольцу №2

- Замените в предыдущей задаче схему «Isend-Recv-Wait» на Sendrecv.