MPI

Динамическое управление процессами

MPI

```
MPI_Comm_spawn(char *command,
char *argv[], int maxprocs,
MPI_Info info, int root,
MPI_Comm comm, MPI_Comm
*intercomm, int
array_of_errcodes[])
```

Попытка породить **maxprocs** процессов, выполняющих программу **command**, которая будет выполнена командным интерпретатором.

MPI

Создаётся интеркоммуникатор **`intercomm`**, объединяющий порождающие и порожденные процессы для возможности дальнейших обменов данными. В **`argv`** передаются параметры запуска программы **`command`** (если не требуется, можно указать предопределённую константу **`MPI_ARGV_NULL`**). Первые четыре аргумента значимы только на процессе **`root`**, на остальных процессах они игнорируются.

MPI

Вызов является коллективным для процессов коммуникатора **`comm`** и не завершается до тех пор, пока во всех порождённых процессах не будет вызвана процедура **`MPI_Init`**. И наоборот, процедура **`MPI_Init`** в порождаемых процессах не выполнится, пока все порождающие процессы не вызовут **`MPI_Comm_spawn`**.

MPI

Порождённые процессы получают свой собственный коммуникатор **`MPI_COMM_WORLD`**, отличный от того, который был у выполнявшихся ранее процессов. Порядок нумерации процессов в группах интеркоммуникатора **`intercomm`** соответствует порядку процессов в коммуникаторе **`comm`** порождающих процессов и в коммуникаторе **`MPI_COMM_WORLD`** порождённых процессов.

MPI

В порождённых процессах интеркоммуникатор может быть получен при помощи вызова процедуры **`MPI_Comm_get_parent`**. Если не удаётся породить **`maxprocs`** процессов, то возвращается ошибка **`MPI_ERR_SPAWN`**. Аргумент **`info`** задаёт информацию о том, где и как запускать порождаемые процессы. Если не требуется, можно использовать предопределённую константу **`MPI_INFO_NULL`**.

MPI

```
int MPI_Comm_get_parent(MPI_Comm *parent)
```

Процедура возвращает на динамически порождённых процессах интеркоммуникатор **parent** для общения с порождающими процессами. Если вызов стоит не в порождённом процессе или если коммуникатор порождающего процесса больше не существует или недоступен, то возвращается значение **MPI_COMM_NULL**.

MPI

master.c:

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char **argv)
{
   int size, rank1, rank2;
   MPI_Status status;
   MPI_Comm intercomm;
   char slave[10]="./slave";
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_spawn(slave, MPI_ARGV_NULL, 2, MPI_INFO_NULL,
0, MPI_COMM_SELF, &intercomm, MPI_ERRCODES_IGNORE);
   MPI_Recv(&rank1, 1, MPI_INT, 0, 0, intercomm, &status);
   MPI_Recv(&rank2, 1, MPI_INT, 1, 1, intercomm, &status);
```

MPI

```
    printf("Slaves %d and %d are working\n", rank1, rank2);
    MPI_Finalize();
    return 0;
}
```

MPI

slave.c:

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char **argv)
{
   int rank, size;
   MPI_Comm intercomm;
   MPI_Init(&argc, &argv);
   MPI_Comm_get_parent(&intercomm);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Send(&rank, 1, MPI_INT, 0, rank, intercomm);
   MPI_Finalize();
   return 0;
}
```