

# Средства и системы параллельного программирования

#12. MPI I/O

# Коллективная операция открытия файла

```
int MPI_File_open(MPI_Comm comm, char *filename, int amode,  
MPI_Info info, MPI_File *fh)
```

`filename` - название открываемого файла

`amode` - режим открытия файла (см. след. слайд)

Filemode и amode должны совпадать для всех процессов!

`info` - дополнительные параметры для файла

`comm` - коммуникатор, в который включены все процессы, которые будут работать с файлом

`fh` - новый инициализированный "MPI-дескриптор" файла, который нужно использовать далее для всех MPI-IO операций.

# Access modes

**MPI\_MODE\_RDONLY**

**MPI\_MODE\_RDWR**

**MPI\_MODE\_WRONLY**

**MPI\_MODE\_CREATE**

**MPI\_MODE\_EXCL**

**MPI\_MODE\_DELETE\_ON\_CLOSE**

**MPI\_MODE\_UNIQUE\_OPEN**

**MPI\_MODE\_SEQUENTIAL**

**MPI\_MODE\_APPEND**

# Коллективная операция закрытия файла

```
int MPI_File_close(MPI_File *fh)
```

Операции MPI-IO в общем случае асинхронны (относительно завершения файловой записи, а не относительно возврата управления в MPI) .

File\_close синхронизирует все ранее инициализированные операции и закрывает файл

Синхронизировать без закрытия - `int MPI_File_sync(MPI_File fh)`

# Как мы представляем файл?

Для каждой позиции в файле у нас есть **displacement** - число байт от начала файла

Для каждой позиции есть также **offset** - число элементов типа **etype** от начала файла

Тип **etype** может быть как стандартным типом (**MPI\_INT**, **MPI\_FLOAT**), так и любым производным

# Операция записи в файл

```
int MPI_File_write_at(MPI_File fh, MPI_Offset offset, void  
*buf, int count, MPI_Datatype datatype, MPI_Status *status)
```

Операция выполняется одним процессом за раз. Начиная с позиции `offset`, из буфера `buf` записываем из `count` элементов типа `datatype` в файл.

## Ещё операции чтения и записи

```
int MPI_File_write(MPI_File fh, void *buf, int count,  
MPI_Datatype datatype, MPI_Status *status)
```

```
int MPI_File_read(MPI_File fh, void *buf, int count,  
MPI_Datatype datatype, MPI_Status *status)
```

По заранее установленным позициям пишем или читаем данные в файл/из файла.

## Установка файлового указателя

```
int MPI_File_seek(MPI_File fh, MPI_Offset offset, int  
whence)
```



## Установить новое представление файла

```
int MPI_File_set_view(MPI_File fh, MPI_Offset disp,  
MPI_Datatype etype, MPI_Datatype filetype, char *datarep,  
MPI_Info info)
```

Ставим новое представление файла в позицию `disp` байт от начала файла и говорим, что новый “ноль файла” – здесь. Более того, говорим, какой тип является элементарным и какими типами мы будем оперировать (`filetype`)

## Коллективная операция записи в файл

```
int MPI_File_write_all(MPI_File fh, const void *buf, int count,  
MPI_Datatype datatype, MPI_Status *status);
```

# Задание

Модифицировать одно из решений домашних заданий 9, 10, 11 на ваш выбор:

Вместо генерации (псевдо)случайных чисел для инициализации начальных данных для работы алгоритма на лету, добавить режим к вашей программе, чтобы данные читались из файла. И обеспечить запись результата в файл. Файл - единый для всех процессов. Входной и выходной файлы - разные.

Если выбрали задание 11 для модификации - читать нужно по одному значению из файла за раз

Задание можно сдать на паре вместе с одной из домашних, а можно прислать модификацию уже сданного задания на github/почту. Отдельно на паре сдавать задание не требуется, отчёт не нужен

Дедлайн: 15.12, 29.12