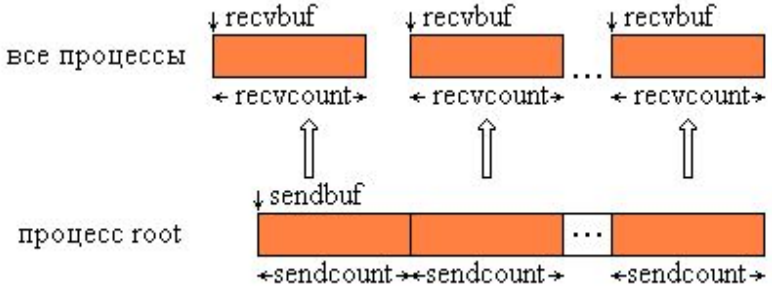
Средства и системы параллельного программирования

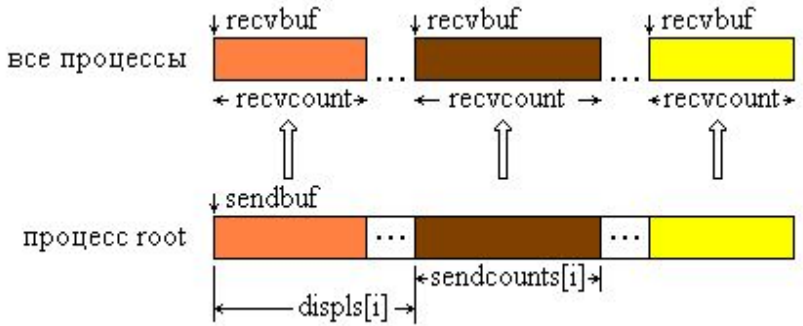
#8. Коллективные операции

MPI Scatter

```
int MPI_Scatter(void* sendbuf, int
sendcount, MPI_Datatype sendtype,
void* recvbuf, int recvcount,
MPI_Datatype recvtype, int root,
MPI_Comm comm)
```



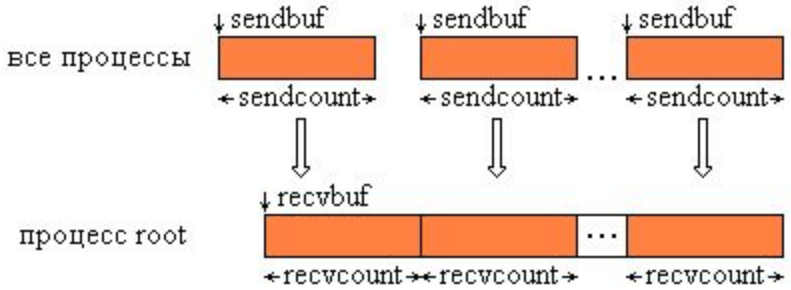

```
int MPI_Scatterv(void* sendbuf,
int *sendcounts, int *displs,
MPI_Datatype sendtype, void*
recvbuf, int recvcount,
MPI_Datatype recvtype, int root,
MPI_Comm comm)
```



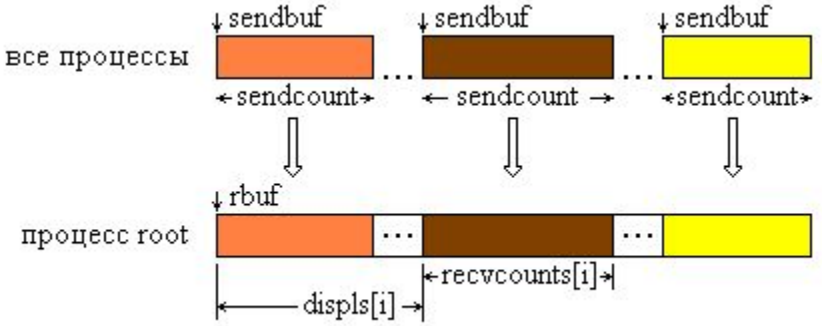
Самое главное - помнить, что root рассылает не только другим в их recvbuf, но и себе в recvbuf.

MPI Gather

```
int MPI_Gather (void* sendbuf,
int sendcount, MPI_Datatype
sendtype,
void* recvbuf, int recvcount,
MPI_Datatype recvtype,
int root, MPI_Comm comm)
```




```
int MPI_Gatherv(void* sendbuf,
int sendcount, MPI_Datatype
sendtype,
void* rbuf, int *recvcounts, int
*displs, MPI_Datatype recvtype,
int root, MPI_Comm comm)
```



Задание

Реализовать программу, которая распределенно проверяет, является ли последовательность целых чисел неубывающей. Если не является, вернуть индекс первого нарушения.

Корневой процесс (можно root = 0) генерирует размер последовательности (N) и саму последовательность, и отправляет её по процессам. Напечатать результат тоже должен root. Коммуникацию между процессами вести только коллективными операциями.

Как удобно решать задачу? Каждый процесс делает вывод про свой кусок данных, затем редукцией (поиск минимума) собираются значения (N + 1 если у процесса всё хорошо, [0, N] если процесс нашёл проблемную позицию у себя). Если внутри блоков всё ОК, root процесс собирает крайние элементы от всех процессов и проверяет, что самый правый элемент процесса i не меньше, чем самый левый у процесса i + 1.

Запуски делать можно локально, выход на Polus в этом задании не требуется. Отчёт тоже не нужен.

Подсчёт сдвигов и размеров для рассылки можно взять из последнего примера семинара

Дедлайн: 17.11, 24.11