```
Index: io-2.tex
===================================================================
--- io-2.tex	(revision 1903)
+++ io-2.tex	(working copy)
@@ -1090,12 +1090,14 @@
 \emph{offsets} &
   & \mpifunc{MPI\_FILE\_WRITE\_AT} & \mpifunc{MPI\_FILE\_WRITE\_AT
\_ALL} \\
 \cline{2-4}
-& \emph{nonblocking} \&
-  & \mpifunc{MPI\_FILE\_IREAD\_AT}  & \mpifunc{MPI\_FILE\_READ\_AT
\_ALL\_BEGIN} \\
-& \emph{split collective}
-  &                            & \mpifunc{MPI\_FILE\_READ\_AT\_ALL
\_END} \\
-& & \mpifunc{MPI\_FILE\_IWRITE\_AT} & \mpifunc{MPI\_FILE\_WRITE\_AT
\_ALL\_BEGIN} \\
-& &                            & \mpifunc{MPI\_FILE\_WRITE\_AT\_ALL
\_END} \\
+& \emph{nonblocking}
+  & \mpifunc{MPI\_FILE\_IREAD\_AT}  & \mpifunc{MPI\_FILE\_IREAD\_AT
\_ALL} \\
+& & \mpifunc{MPI\_FILE\_IWRITE\_AT} & \mpifunc{MPI\_FILE\_IWRITE\_AT
\_ALL} \\
+\cline{2-4}
+& \emph{split collective} & {N/A} & \mpifunc{MPI\_FILE\_READ\_AT\_ALL
\_BEGIN} \\
+& & & \mpifunc{MPI\_FILE\_READ\_AT\_ALL\_END} \\
+& &                                 & \mpifunc{MPI\_FILE\_WRITE\_AT
\_ALL\_BEGIN} \\
+& &                                 & \mpifunc{MPI\_FILE\_WRITE\_AT
\_ALL\_END} \\
 \hline %-------------------------------------------------------------
 \emph{individual}    & \emph{blocking}
   & \mpifunc{MPI\_FILE\_READ}   & \mpifunc{MPI\_FILE\_READ\_ALL} \\
@@ -1102,12 +1104,14 @@
 \emph{file pointers} &
   & \mpifunc{MPI\_FILE\_WRITE}  & \mpifunc{MPI\_FILE\_WRITE\_ALL} \\
 \cline{2-4}
-& \emph{nonblocking} \&
-  & \mpifunc{MPI\_FILE\_IREAD}  & \mpifunc{MPI\_FILE\_READ\_ALL
\_BEGIN} \\
-& \emph{split collective}
-  &                      & \mpifunc{MPI\_FILE\_READ\_ALL\_END} \\
-& & \mpifunc{MPI\_FILE\_IWRITE} & \mpifunc{MPI\_FILE\_WRITE\_ALL
\_BEGIN} \\
-& &                      & \mpifunc{MPI\_FILE\_WRITE\_ALL\_END} \\
+& \emph{nonblocking}
+  & \mpifunc{MPI\_FILE\_IREAD}  & \mpifunc{MPI\_FILE\_IREAD\_ALL} \\
+& & \mpifunc{MPI\_FILE\_IWRITE} & \mpifunc{MPI\_FILE\_IWRITE\_ALL} \\
+\cline{2-4}
+& \emph{split collective} & {N/A} & \mpifunc{MPI\_FILE\_READ\_ALL
\_BEGIN} \\
+& & & \mpifunc{MPI\_FILE\_READ\_ALL\_END} \\
+& &                                 & \mpifunc{MPI\_FILE\_WRITE\_ALL
\_BEGIN} \\
+& &                                 & \mpifunc{MPI\_FILE\_WRITE\_ALL
\_END} \\
 \hline %-------------------------------------------------------------
 \emph{shared}       & \emph{blocking}
   & \mpifunc{MPI\_FILE\_READ\_SHARED}   & \mpifunc{MPI\_FILE\_READ
\_ORDERED} \\
@@ -1114,12 +1118,14 @@
 \emph{file pointer} &
   & \mpifunc{MPI\_FILE\_WRITE\_SHARED}  & \mpifunc{MPI\_FILE\_WRITE
\_ORDERED} \\
 \cline{2-4}
-& \emph{nonblocking} \&
-  & \mpifunc{MPI\_FILE\_IREAD\_SHARED}  & \mpifunc{MPI\_FILE\_READ
\_ORDERED\_BEGIN} \\
-& \emph{split collective}
-  &                          & \mpifunc{MPI\_FILE\_READ\_ORDERED
\_END} \\
-& & \mpifunc{MPI\_FILE\_IWRITE\_SHARED} & \mpifunc{MPI\_FILE\_WRITE
\_ORDERED\_BEGIN}\\
-& &                          & \mpifunc{MPI\_FILE\_WRITE\_ORDERED
\_END} \\
+& \emph{nonblocking}
+  & \mpifunc{MPI\_FILE\_IREAD\_SHARED}  & {N/A} \\
+& & \mpifunc{MPI\_FILE\_IWRITE\_SHARED} & \\
+\cline{2-4}
+& \it split collective & {N/A} & \mpifunc{MPI\_FILE\_READ\_ORDERED
\_BEGIN} \\
+& & & \mpifunc{MPI\_FILE\_READ\_ORDERED\_END} \\
+& &                                 & \mpifunc{MPI\_FILE\_WRITE
\_ORDERED\_BEGIN} \\
+& &                                 & \mpifunc{MPI\_FILE\_WRITE
\_ORDERED\_END} \\
 \hline
 \end{tabular}
 \end{center}
@@ -1458,6 +1464,28 @@
 \mpifunc{MPI\_FILE\_IREAD\_AT} is a nonblocking version
 of the \mpifunc{MPI\_FILE\_READ\_AT} interface.

+\begin{funcdef}{MPI\_FILE\_IREAD\_AT\_ALL(fh, offset, buf, count,
datatype, request)}
+\funcarg{\IN}{fh}{file handle (handle)}
+\funcarg{\IN}{offset}{file offset (integer)}
+\funcarg{\OUT}{buf}{initial address of buffer (choice)}
```

```
+\funcarg{\IN}{count}{number of elements in buffer (integer)}
+\funcarg{\IN}{datatype}{datatype of each buffer element (handle)}
+\funcarg{\OUT}{request}{request object (handle)}
+\end{funcdef}
+
+\cdeclindex{MPI\_Request}%
+\cdeclindex{MPI\_File}%
+\cdeclindex{MPI\_Offset}%
+\mpibind{MPI\_File\_iread\_at\_all(MPI\_File~fh, MPI\_Offset~offset,
void~*buf, int~count, MPI\_Datatype~datatype, MPI\_Request~*request)}
+\mpifnewbind{MPI\_File\_iread\_at\_all(fh, offset, buf, count,
datatype, request, ierror) BIND(C) \fargs TYPE(MPI\_File),
INTENT(IN) :: fh \\ INTEGER(KIND=MPI\_OFFSET\_KIND), INTENT(IN) ::
offset \\ TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf \\ INTEGER,
INTENT(IN) :: count \\ TYPE(MPI\_Datatype), INTENT(IN) :: datatype \\
TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
+\mpifbind{MPI\_FILE\_IREAD\_AT\_ALL(FH, OFFSET, BUF, COUNT, DATATYPE,
REQUEST, IERROR)\fargs <type> BUF(*) \\ INTEGER FH, COUNT, DATATYPE,
REQUEST, IERROR \\ INTEGER(KIND=MPI\_OFFSET\_KIND) OFFSET}
+
+\mpifunc{MPI\_FILE\_IREAD\_AT\_ALL} is a nonblocking version of
+\mpifunc{MPI\_FILE\_READ\_AT\_ALL}. See
+Section~\ref{sec:io-semantics-nb-collective},
+page~\pageref{sec:io-semantics-nb-collective} for semantics of
nonblocking
+collective file operations.
+
 \begin{funcdef}{MPI\_FILE\_IWRITE\_AT(fh, offset, buf, count,
datatype, request)}
 \funcarg{\INOUT}{fh}{file handle (handle)}
 \funcarg{\IN}{offset}{file offset (integer)}
@@ -1478,6 +1506,25 @@
 \mpifunc{MPI\_FILE\_IWRITE\_AT} is a nonblocking version
 of the \mpifunc{MPI\_FILE\_WRITE\_AT} interface.

+\begin{funcdef}{MPI\_FILE\_IWRITE\_AT\_ALL(fh, offset, buf, count,
datatype, request)}
+\funcarg{\INOUT}{fh}{file handle (handle)}
+\funcarg{\IN}{offset}{file offset (integer)}
+\funcarg{\IN}{buf}{initial address of buffer (choice)}
+\funcarg{\IN}{count}{number of elements in buffer (integer)}
+\funcarg{\IN}{datatype}{datatype of each buffer element (handle)}
+\funcarg{\OUT}{request}{request object (handle)}
+\end{funcdef}
+
+\cdeclindex{MPI\_Request}%
+\cdeclindex{MPI\_File}%
+\cdeclindex{MPI\_Offset}%
+\mpibind{MPI\_File\_iwrite\_at\_all(MPI\_File~fh, MPI\_Offset~offset,
const void~*buf, int~count, MPI\_Datatype~datatype, MPI
\_Request~*request)}
+\mpifnewbind{MPI\_File\_iwrite\_at\_all(fh, offset, buf, count,
datatype, request, ierror) BIND(C) \fargs TYPE(MPI\_File),
INTENT(IN) :: fh \\ INTEGER(KIND=MPI\_OFFSET\_KIND), INTENT(IN) ::
offset \\ TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf \\
INTEGER, INTENT(IN) :: count \\ TYPE(MPI\_Datatype), INTENT(IN) ::
datatype \\ TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
+\mpifbind{MPI\_FILE\_IWRITE\_AT\_ALL(FH, OFFSET, BUF, COUNT,
DATATYPE, REQUEST, IERROR)\fargs <type> BUF(*) \\ INTEGER FH, COUNT,
DATATYPE, REQUEST, IERROR \\ INTEGER(KIND=MPI\_OFFSET\_KIND) OFFSET}
+
+\mpifunc{MPI\_FILE\_IWRITE\_AT\_ALL} is a nonblocking version of
+\mpifunc{MPI\_FILE\_WRITE\_AT\_ALL}.
+
 \subsection{Data Access with Individual File Pointers}
 %-----------------------------------------------------
 \label{sec:io-indiv-ptr}
@@ -1691,6 +1738,23 @@
 \end{verbatim}
 \end{example}

+\begin{funcdef}{MPI\_FILE\_IREAD\_ALL(fh, buf, count, datatype,
request)}
+\funcarg{\INOUT}{fh}{file handle (handle)}
+\funcarg{\OUT}{buf}{initial address of buffer (choice)}
+\funcarg{\IN}{count}{number of elements in buffer (integer)}
+\funcarg{\IN}{datatype}{datatype of each buffer element (handle)}
+\funcarg{\OUT}{request}{request object (handle)}
+\end{funcdef}
+
+\cdeclindex{MPI\_Request}%
+\cdeclindex{MPI\_File}%
+\mpibind{MPI\_File\_iread\_all(MPI\_File~fh, void~*buf, int~count,
MPI\_Datatype~datatype, MPI\_Request~*request)}
+\mpifnewbind{MPI\_File\_iread\_all(fh, buf, count, datatype, request,
ierror) BIND(C) \fargs TYPE(MPI\_File), INTENT(IN) :: fh \\ TYPE(*),
DIMENSION(..), ASYNCHRONOUS :: buf \\ INTEGER, INTENT(IN) :: count \\
TYPE(MPI\_Datatype), INTENT(IN) :: datatype \\ TYPE(MPI\_Request),
INTENT(OUT) :: request \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifbind{MPI\_FILE\_IREAD\_ALL(FH, BUF, COUNT, DATATYPE, REQUEST,
IERROR) \fargs <type> BUF(*) \\ INTEGER FH, COUNT, DATATYPE, REQUEST,
IERROR}
+
+\mpifunc{MPI\_FILE\_IREAD\_ALL} is a nonblocking version
+of \mpifunc{MPI\_FILE\_READ\_ALL}.
+
 \begin{funcdef}{MPI\_FILE\_IWRITE(fh, buf, count, datatype, request)}
 \funcarg{\INOUT}{fh}{file handle (handle)}
 \funcarg{\IN}{buf}{initial address of buffer (choice)}
@@ -1708,6 +1772,23 @@

 \mpifunc{MPI\_FILE\_IWRITE} is a nonblocking version of the
\mpifunc{MPI\_FILE\_WRITE} interface.

+\begin{funcdef}{MPI\_FILE\_IWRITE\_ALL(fh, buf, count, datatype,
request)}
+\funcarg{\INOUT}{fh}{file handle (handle)}
+\funcarg{\IN}{buf}{initial address of buffer (choice)}
+\funcarg{\IN}{count}{number of elements in buffer (integer)}
+\funcarg{\IN}{datatype}{datatype of each buffer element (handle)}
+\funcarg{\OUT}{request}{request object (handle)}
+\end{funcdef}
+
+\cdeclindex{MPI\_Request}%
+\cdeclindex{MPI\_File}%
+\mpibind{MPI\_File\_iwrite\_all(MPI\_File~fh, const void~*buf,
int~count, MPI\_Datatype~datatype, MPI\_Request~*request)}
+\mpifnewbind{MPI\_File\_iwrite\_all(fh, buf, count, datatype,
request, ierror) BIND(C) \fargs TYPE(MPI\_File), INTENT(IN) :: fh \\
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf \\ INTEGER,
INTENT(IN) :: count \\ TYPE(MPI\_Datatype), INTENT(IN) :: datatype \\
TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
+\mpifbind{MPI\_FILE\_IWRITE\_ALL(FH, BUF, COUNT, DATATYPE, REQUEST,
IERROR)\fargs <type> BUF(*) \\ INTEGER FH, COUNT, DATATYPE, REQUEST,
IERROR}
+
+\mpifunc{MPI\_FILE\_IWRITE\_ALL} is a nonblocking version
+of \mpifunc{MPI\_FILE\_WRITE\_ALL}.
+
 \begin{funcdef}{MPI\_FILE\_SEEK(fh, offset, whence)}
 \funcarg{\INOUT}{fh}{file handle (handle)}
 \funcarg{\IN}{offset}{file offset (integer)}
@@ -3339,6 +3420,37 @@
 Different processes can pass different values for other arguments
 of a collective routine unless specified otherwise.

+\subsection{Nonblocking Collective File Operations}
+%-------------------------------------
+\label{sec:io-semantics-nb-collective}
+Nonblocking collective file operations are defined only for data
access
+routines with explicit offsets and individual file pointers but not
with
+shared file pointers.
+
+Nonblocking collective file operations are subject to the same
restrictions as
```

```
+blocking collective I/O operations. All processes belonging to the
group of
+the communicator that was used to open the file must call collective
I/O
+operations (blocking and nonblocking) in the same order. This is
consistent
+with the ordering rules for collective operations in threaded
environments.
+For a complete discussion, please refer to the semantics set forth in
+Section~\ref{coll:correct} on page~\pageref{coll:correct}.
+
+Nonblocking collective I/O operations do not match with blocking
collective
+I/O operations. Multiple nonblocking collective I/O operations can be
+outstanding on a single file handle.  High quality MPI
implementations should
+be able to support a large number of pending nonblocking I/O
operations.
+
+All nonblocking collective I/O calls are local and return
immediately,
+irrespective of the status of other processes. The call initiates the
+operation which may progress independently of any communication,
computation,
+or I/O. The call returns a request handle, which must be passed to a
+completion call. Input buffers should not be modified and output
buffers
+should not be accessed before the completion call returns. The same
progress rules
+described for nonblocking collective operations apply for nonblocking
+collective I/O operations. For a complete discussion, please refer to
the
+semantics set forth in Section~\ref{sec:nbcoll} on
+page~\pageref{sec:nbcoll}.
+
 \subsection{Type Matching}
 %------------------------

@@ -3750,6 +3862,21 @@
 %%SKIP
 %%ENDHEADER
 \begin{verbatim}
+MPI_File_iwrite_all(fh,...) ;
+MPI_File_iread_all(fh,...) ;
+MPI_Waitall(...) ;
+\end{verbatim}
+
+In addition, as mentioned in Section~\ref{sec:io-semantics-nb-
collective} on
+page~\pageref{sec:io-semantics-nb-collective}, nonblocking collective
```

```
I/O
+operations have to be called in the same order on the file handle by
all
+processes.
+
+Similar considerations apply to conflicting accesses of the form:
+%%HEADER
+%%SKIP
+%%ENDHEADER
+\begin{verbatim}
 MPI_File_write_all_begin(fh,...) ;
 MPI_File_iread(fh,...) ;
 MPI_Wait(fh,...) ;
```