Published: 8/31/66

Identification

Sequential Logical Record I/O
J. F. Ossanna, V. A. Vyssotsky, G. G. Ziegler

Purpose

The Multics I/O system provides capability for sequential record I/O. This section describes in detail the I/O system calls for performing sequential logical record I/O.

Sequential Logical Record Frames

An existing logical record frame may be attached to a process (or a new logical record frame may be created and attached to a process) as a sequential frame by an attach call to the I/O system. A frame which already exists when it is so attached may have previously been attached to processes as either a sequential or a random frame or both, but not both at the same time. When attached as a sequential frame, a logical record frame may be regarded as a sequence of records; however, some of the records may be missing (see section BF.1.21 for a discussion of status returned on requests for missing records). The primitive operations available for transmitting data to and from a serial logical record frame are described in the following discussion.

The Write Call

Suppose a new sequential logical record frame has been created and attached to an inout stream named alpha. Data may be placed in the frame by using a write call. The general form of a write call is:

```
   call write(name,recno,workspace,nelem[,status])
```

The argument name is a character string of 1 to 31 characters. Its content is either a streamname or a frame id. If name is a streamname, it refers to the frame to which the stream is attached. The argument recno is a 35 bit signed integer whose value must be non-negative. The value of recno is the difference between the record number of the record to be written and the current record number. The argument workspace is a pointer to the data to be written. Specifically, in PL/I terminology, workspace is a pointer variable; the I/O system will act as if the based variable associated with the pointer variable workspace were a bit string of length nelem times the element size. The argument nelem is a 35 bit signed integer specifying the number of elements to be written. The value of nelem must be in the range $0 \leq \text{nelem} \leq N$, where N is the declared maximum record size of the frame. The optional argument *status* is a bit string returned by the I/O system to the caller, containing status information about the transaction.

After a record has been written by a *write* call, another *write* call with recno = 0 or null will cause the next record to be written. In normal *truncation* mode for writing of sequential logical record frames, writing a record causes loss of all records with higher record numbers. For example, if a frame containing ten records of 1000 elements each is attached as a sequential logical record frame and connected to stream alpha, then after

```
call write('alpha',0,data,100)
```

the frame consists of one record, containing 100 elements from workspace area data. (See section BF.1. for a discussion of the *truncation* and *replacement* modes.)

The Read Call

Records, or parts of records, may be read by the *read* call, whose general form is

```
call read(name,recno,workspace,nelem,[,nelmt[,status]])
```

The arguments of *read* are the same as the corresponding arguments of *write*, except that the *read* call has an additional optional argument, *nelmt*. This argument is a 35 bit signed integer, returned by the I/O system. Its value is the number of elements transmitted from the frame to the caller's workspace, and lies in the range $0 \leq \text{nelmt} \leq \text{nelem}$. If no data is read, nelmt = 0. If the read request is completely fulfilled, nelmt = nelem. If the record to be read is less than nelem elements long, the record is transmitted, and the value of nelmt is the length of the record, in elements. As an example, suppose that the frame attached to stream alpha is positioned at the beginning of frame, either immediately after attachment or immediately after a *first* call. Suppose further that the element size is 36 bits and that the first record is 100 elements (words) long. Then the call

```
call read('alpha',0,data,100)
```

will cause the first record of the frame to become the current record, and would cause the record to be read into the first 100 words of data area *data*. If a read call requests data from a valid record, but from a place partly or wholly beyond the actual end of the record, any of the requested data which exists will be transmitted, and the status return will show that less data was transmitted than had been requested. For example, if the record of the previous example had been read by

```
call read('alpha',0,data,200,amount,state)
```

then the record is read into the first 100 words of data, words 101-200 of data are unchanged, state indicates less data read than was called for, and the value of amount is 100.

The Current Record

After a successful write call, the next record to be written is known as the current record. In most circumstances a frame connected as a sequential logical record frame has a current record. Specifically, a sequential logical record frame has a current record under the following circumstances. If the most recent read, write, seek or delete call referenced a record of the frame (i.e. was not rejected by the I/O system) and if no subsequent first or tail call has occurred, then the record just beyond the last record referenced by that most recent read, write, seek or delete call is the current record. This is true even if the last record contains no data (e.g. has just been deleted).

After a frame is initially attached or immediately after a first call, the current record is the first record. After a tail call the current record is LAST+1. For a precise definition of current record, see section BF.1.10.

The Tell Call

It is often useful to be able to determine the current record number of a logical record frame. This can be accomplished by means of the tell call, whose general form is:

```
call tell(name,recno[,status])
```

The arguments name and status are the same as the corresponding arguments of a write call. The argument recno is a 35 bit signed integer. The value of recno at time of call will be ignored and overwritten by the I/O system. At time of return, recno will contain the current record number for the indicated frame, unless the call was rejected by the I/O system (bit 4 or bit 15 of status set to 1).

A call to tell does not change the current record number, nor does it change the data content of the frame. On return from a call to tell the value of the argument status will be exactly what it would have been if the call had been a call to seek with recno = 0.

The Seek Call

The seek call allows a record to be designated as the current record without reading or altering the contents of the record. Its general form is

```
call seek(name,recno[,state])
```

and its arguments are the same as the corresponding arguments of write. If stream alpha is connected to a frame which has a current record, then

```
call seek('alpha',n)
```

will cause the current record to be n records after the record which was the current record before the call. For a precise definition of current record see section BF.1.10.

In reading a frame, the seek call may be used to skip over a record or records. For example, if the frame attached to a stream alpha has just been positioned by a first call, the sequence

```
call read('alpha',,data,50)
call seek('alpha',1)
call read('alpha',0,other,50)
```

will read 50 words from the first record into data, will skip the second record, and will read 50 words from the third record into other.

The Delete Call

The general form of the delete call is

```
call delete(name,recno[,status])
```

and its arguments have the same form as the corresponding arguments of the write call. If a delete call is given for a logical record frame in the normal truncation mode, the specified record and all following records are deleted. If a logical record frame is in the replacement mode it is sometimes desirable to be able to delete a record completely without rewriting it. This can be done with the delete call.

After a record has been deleted, a subsequent read call for that record will give status return showing that the record does not exist. If the content of a record is destroyed by a call to write the record with nelem = 0, a subsequent read call for that record will give status return showing that the record existed and that its length was 0. This distinction may, of course, be ignored by a calling procedure if the difference is irrelevant.

The First Call

After reading or writing part or all of a sequential frame, it is frequently necessary to go back to the beginning of the frame and start reading or writing again from the beginning. This is accomplished by the first call, whose general form is

```
call first(name[,status])
```

For example, suppose that a number of records have been written in a frame connected to stream alpha, and it is now desired to read those records.

```
call first('alpha')
```

will position the frame so that a read call can read data from the first record of the frame.

The Tail Call

When adding records to an existing logical record frame it is useful to be able to skip to the end of the current records. This may be done with the tail call whose general form is

```
call tail(name[,status])
```

Following such a call the current record number is LAST+1 and a write with recno = 0 or null would write data immediately following the last record already in the frame. A read call of any kind would get a status return showing end-of-frame.