To: Distribution

From: Mike Asherman

Date: 10/22/75

Subject: Proposed vfile_ Extensions for Fortran/Basic I/O

Introduction

Several extensions to vfile_ are proposed, including a new file type, several new operations and attach options, and various other changes to the current specs. The primary motivation for making these changes in vfile_ is to provide interchangeable support for Fortran/Basic I/O in FAST. The proposed changes also serve to fill a logical gap in the existing file I/O system: the lack of a simple direct-accessing capability. Some minor alterations for the sake of user convenience are also included in this proposal.

New File Type

The most significant extension contemplated is a new file type--the "blocked" file. A distinct header would be used to keep track of each file's length (records) and characteristic maximum record size (bytes). The file is divided into blocks of fixed length, each with a single-word header specifying the length (bytes) of the contained record. A structure of this type would permit very fast direct access to variable-length records on the numerical record position, (0,1,...). Such files are required in Basic (random string files) and in Fortran (random string and binary files).

To explicitly attach a blocked file, the -blocked -n- option should be used. The permissible opening modes are the same as those for sequential files. A new position operation (type=2) would permit direct access to a record whose "absolute" position is specified; another operation, control ("read_position"), would return the numerical position of the next record in the file, as well as the end-of-file position. Two other operations are also proposed: control ("max_rec_len"), for changing/obtaining the maximum record size of an empty file; control ("truncate"), for truncation at the next record position.

In order to facilitate Fortran/Basic use of a single file-position, the write operation could be supported in sequential_update openings, with the desired effect of either appending to the file or replacing the next record, depending on the next record position.

The delete operation would not be supported with blocked files.

New Attach Options

The following new attach options are proposed:

1. -blocked -n- specifies attachment to a blocked file. If a non-empty file exists, n is ignored and may be omitted. Otherwise, n is used to set the maximum record size (bytes).

2. -no_trunc indicates that a put_chars operation into the middle of an unstructured file (stream_input_output) is permitted, and no truncation is to occur in such cases. Also prevents the truncation of an existing file at open. (This option is provided to support Fortran "direct" files; in conjunction with the -header option, it supports Fortran/Basic random numeric files.)

3. -header -n- for use with unstructured files, this option indicates that a header is expected in an existing file, or is to be created for a new file. If a header is specified, it contains an optional identifying number, which effectively permits user-defined file types. If n is given and the file exists, the file identifier must be equal to n; a new file takes the value of n, if given, as its identifier. The header is maintained and becomes invisible only with the explicit use of this option.

4. -append in _input_output openings, this causes put_chars and write_record operations to add to end_of_file instead of truncating when the file position is not at end_of_file. Also the position is initially set to beginning_of_file, and an existing file is not truncated at open.

5. -must_exist — indicates that a new file is not to be created if an attempt is made to open a non-existing file for output, input-output, or update.

New Operations

The following new operations are proposed:

control operations:

1. "read_position" — this order is accepted when the I/O switch is open and attached to a non-indexed file. The operation returns the ordinal position (0,1,2,...) of the next record (byte for unstructured files), and that of the end of file, relative to the file base. The file base is just beyond the header, if a header is present. If the next position is undefined, the code error_table_$no_record is returned, but the end_of_file position is always returned.

   For this order, the info_ptr argument must point to a structure of the following form:

```
dcl  1 info based(info_ptr),
       2 next_position fixed(30), /*output*/
       2 last_position fixed(30); /*output*/
```

2. "truncate" — this order is accepted when the I/O switch is attached to a non-indexed file open for _input_output or _update. The operation truncates the file at the next record (byte for unstructured files). If the next position is undefined, the code error_table_$no_record is returned.

   No info structure is required for this order.

3. "max_rec_len" — this order is accepted when the I/O switch is open and attached to a blocked file. The operation returns the maximum record length (bytes) of the file. A new maximum length can be set by specifying a non-zero value for a second argument. In this case the file must empty and open for modification, or the code error_table_$no_operation is returned.

For this order the info_ptr argument must point to a structure of the following form:

```
dcl  1 info based(info_ptr),
       2 old_max_recl fixed, /*output*/
       2 new_max_recl fixed; /*input*/
```

other operations not currently supported:

4. position (2,n) specifies "absolute" positioning directly to the nth byte or record of a file (non-indexed), beyond the header if present.

5. write_record (in sequential_update openings). For adding or replacing a record at the next record position (only rewrite is currently supported in this mode). Applies only to blocked and sequential files.

Other Proposed Changes

The vfile_status command should recognize the new file type and provide more information about non-indexed files; a subroutine counterpart to the command returning a structure appropriate to the file type would also be desirable.

Some other changes related to user convenience are being considered:

1. Openings for update create file if none exists (unless -must_exists attachment).

2. ignore -extend option in openings for input or update (currently this is treated as an error).

3. don't truncate existing files on opening for output or inout_output without -extend if the safety switch is on.

4. redefine specs with regard to the current record position to state that the current position is always set to the next record position immediately after a position operation, and at opening.