To: Distribution

From: Janice B. Phillipps

Date: 07/08/75

Subject: Design of tape_in, tape_out Enhancements

INTRODUCTION

Currently, the tape_in and tape_out commands work with an ANSI/IBM write-around using nstd_ thru ios_ and a pre-installation version of the recuction compiler. They only support transactions between unlabeled 9-track tapes and unstructured files in the storage system. Proposed here are changes to be made to tape_in and tape_out make them standard for service installation and to enhance their usefulness as part of the Multics tape facility. Attached to this MTB is MPM draft documentation of the enhanced tape_in and tape_out commands.

The changes planned for the tape_in and tape_out commands include the following: substituting the use of tape_ansi_, tape_ibm_, tape_mult_ and ntape_ using iox_ for the use of nstd_ thru ios_; updating tape_in and tape_out to use the installed version of the reduction_compiler; supporting both labeled as well as unlabeled tapes; supporting sequential format as well as unstructured format files in the storage system; having default conditions for for each IO module supported by tape_in and tape_out as well as having a default tape IO module for tape_in and tape_out.

No immediate plans are made to handle future RCP changes that might be required, and no plans have been made to accommodate 7-track tapes.

The enhancement of tape_in and tape_out implementation schedule, in outline, follows.

1. Update tape_in and tape_out to work with the installed version of the reduction_compiler.

2. Update tape_in and tape_out to work with tape_ansi and tape_ibm IO modules only handling unstructured files in storage system. Tapes both labeled and unlabeled will be supported.

3. Update tape_in and tape_out to support transfer of sequential format files in the storage system.

4. Update tape_in and tape_out to use tape_mult_ and ntape_ IO modules for sequential format and unstructured format file transfer.

IMPLEMENTATION

The commands tape_in and tape_out input an intermediary form of the control file, written by the user, into the reduction_compiler such that the resulting compiled code can be executed in order to accomplish a file transfer from storage to tape or vice versa.

The tape_in, tape_out user must specify in the control file the specific IO module which he wishes to perform his tape processing as the design scope of tape_in and tape_out does not encompass a search for the most appropiate IO module for a given input control file, as has been suggested in the design discussions of a "superdim_". The storage system files are all referenced alike thru vfile_, so the user need not specify here; however, the storage system file format has to be specified by the user in the control file as there no present way to get this information elsewhere short of getting it from an error reported for the wrong choice of format.

tape_in, tape_out sets up for attaching and opening each file read from or written onto tape by parsing the control file supplied by the user. The user is informed that the tape IO module he selects determines tha possible options available for tape precessing. tape_in and tape_out know what each IO module regards as valid attachment options and puts together an acceptable attach_description for the selected IO module.

Tape file positioning is only implemented by the tape_ansi_ and tape_ibm_ IO modules. tape_in and tape_out will accomplish tape file positioning thru the IO module by file name, or by file name and file number in combination, whatever is specified in the control file. As ntape_ and tape_mult_ make attachments with reference to the beginning of the tape and offer no tape control options, no automatic tape positioning will be available to the tape_in, tape_out user using one of these IO modules.

IO error reporting (run-time) will consist of passing along to the user errors returned from the IO module in use. A special case will be implemented for the error_table_$tape_error message returned by tape_ansi_ or tape_ibm_. In this case, the tape_in and tape_out commands will issue a status control call to return the IOM status string interpretation of major and all minor status to the user. Otherwise, as for other IO modules, only the status returned will be passed to the user.

The file transfers between tape and storage system sequential files will implemented with pl/1 I/O. One file system record will correspond to one tape file record.

DESIGN ISSUES

The philosophy behind implementation of tape_in, tape_out has been to support the various tape IO module features as far as is possible while keeping tape_in and tape_out easy and convenient to use.

The present design for the tape_in, tape_out commands specifies a command line option for use with tape_ansi_ and tape_ibm_ "-retain", which needs to be taken up here. This discussion becomes academic when the full RCP package is installed, as the retension and release of IO module resourses will then be up to the user. However, that is a ways away, and it would be nice to have an interim support of a facility that is already offered the user if he use the tape_ansi_ or tape_ibm_ IO modules directly.

The -retain option allows the user to request that the IO module resources (tape volumes and tape drives) remain assigned to his process accross file attachements made on a given volume set. Upon completing the execution of a <volume_set-group> in a control file, tape_in, tape_out sees to it that the IO module resources are unassigned. The current default resource disposition for the tape_in, tape_out commands, is the same as that implemented for tape_ansi_ and tape_ibm_: all tape resources are unassigned upon file detachment, i.e. when a tape_in, tape_out control file <file-group> is finished executing. The plan for default resource disposition seems statisfactory, however the design for the resource retension falls short of being really functional and departs from the features offered by the IO modules tape_ansi_ and tape_ibm_.

If a user wants to write a tape according to a given control file and then he wants to read that tape using the same or a similar control file, it is undesirable to have the volumes and drives assigned to his process for the write, become unassigned and then assigned again for the read. The IO modules tape_ansi_ and tape_ibm_ provide the user with a control option which allows him to change the resource disposition at any time within a given process. The user of tape_in, tape_out cannot make direct use of this control operation as things stand now; the user must give up resources upon finishing a <volume_set-group>. It would be advantagous for a user to be able to hold the resources through a process if there was a good way of getting rid of them when they were no longer needed, short of doing a new_proc. One solution to this situation would be for tape_in and tape_out to maintain a table of which IO modules made which retain attachments and then search thru the table to release the resources upon completion of a control file. This would mean however, that in order to read then write a given volume set without giving up the drive(s), the user would have to set up one control file to do the work, and could not reuse the same control file without first giving up the drive(s). Proposed here as an interim solution to this problem until RCP takes care of it, is to have an entry in each IO module which the user could call directly to change the resource disposition at any time during his process.

FURTHER ENHANCEMENTS BEING CONSIDERED

Currently, questions posed by tape_ansi_ and tape_ibm_ as to expiration dates and continuing volumes, are written on the stream user_i/o. There could be a control file statement available which would take the pathname of a user written command question handler which would be called to answer questions posed by tape_ansi_ or tape_ibm_.

If, once the resource disposition of an IO module were set for a process, so that a volume set could be mounted and the same control file could be used to both read and write on that volume set, then it would be useful to implement a "-protect" control argument for the tape_in tape_out command line. The "-protect" command line option would specify for a volume set to be mounted with rings but hardware file protect be set to prevent spurious writing. In this way a volume set would not have to be taken down to insert tape rings between the read and write operations, thus maintaining the idea of retain. tape_ansi_ and tape_ibm_ take advantage of this protect option directly. Right now, with the current design of resource dispositions, it does not make sense to implement the -protect feature.

<u>Name</u>: tape_in, tin
tape_out, tout

These commands allow the user to transfer files between the storage system and magnetic tape. tape_in reads from tape to the storage system; tape_out writes from the storage system to tape. Both unstructured and sequential format storage system files are supported; both labeled and unlabeled 9-track tapes may be read or written.

The tape_in and tape_out commands access one of four system IO modules thru the I/O system to accomplish file transfer. The user of tape_in and tape_out first decides which IO module he wishes use to do the file transfer based on which tapes he wishes to process (ANSI standard, IBM standard, Multics standard, or non-standard), based on whether the tapes are or are to be labeled or unlabeled, and based on which tape file attributes he will find suitable. See MPM write-ups on tape_ansi_, tape_ibm_, ntape_, and tape_mult_ for background on the IO modules; and in particular, refer to tape_ansi_ or tape_ibm_ section <u>Definition of Terms</u> for the definitions of record, block, file, volume, file set and volume set as they are used below.

<u>Usage</u>: tape_in pathname -control_arg<u>1</u>- ... -control_arg<u>n</u>-
tape_out pathname -control_arg<u>1</u>- ... -control_arg<u>n</u>-

1) pathname	is the path name of the control file which governs the file transfer. If pathname does not end with the suffix .tcl, .tcl will be supplied.
2) -severity<u>i</u>, -sv<u>i</u>	causes the tape_in, tape_out compiler error messages with severity less than <u>i</u> (where <u>i</u> is 0, 1, 2, 3, or 4) to not be written into the "error_output" IO stream. The default value for <u>i</u> is 0.
3) -brief, -bf	causes the error messages from the tape_in, tape_out compiler to be written onto the stream "error_output" in brief

mode. The brief message has the same format as the normal, non-brief message, but uses the short form of the error_table_ entries.

For further information on error reporting see below under the heading Error Diagnostics.

4) -retain, -ret	specifies that the IO module's resources (tape drives and tape volumes) remain assigned to the user's process throughout the execution of each <volume_set-group> in the control file. The ntape_ and tape_mult_ IO modules do not have the retain capability, therefore this control argument only has meaning when used in conjunction with the tape_ansi_ or tape_ibm_ IO modules. If used in conjunction with any other tape IO module, this control argument will be ignored. If several files on a ANSI or IBM tape volume set are to be read or written, use of the -retain option can save the user's process from having to compete with other processes for device assignment each time a file on the volume set is referenced. The default resource disposition (applies to all IO modules) unassigns all devices and volumes with every control file <file-group> completion (applies to all IO modules).
5) -force, -fc	specifies that the expiration date of a tape file to be overwritten is to be ignored. This option extends unconditional permission to overwrite a tape file, regardless of the file's "unexpired" status. If the the -force option is used with the tape_in command, an error is indicated. This control argument only has meaning when used in conjuction with the tape_ansi_ or tape_ibm_ IO modules. If used with other tape IO modules, the control argument is ignored.

THE TCL PROGRAM

The control file which governs the file transfer is actually a program, written by the user, in the tape_in/out control language (tcl). The contents of this control file specifies the attributes of the file(s) in the storage system to be copied or created, and the attributes of the file(s) on the tape volume(s) to be copied or created, and the name of the IO module to do the actual file transfer(s). When the user issues the tape_in or tape_out command, the control file specified by name in the command line (pathname) is compiled and if the compilation is successful, the generated object code is then executed in order to accomplish the desired file transfer(s).

A tcl program consists of 1 or more <volume_set-group>s. A <volume_set-group> is a series of statements which specifies the file transfer(s) to be performed between the storage system and a particular tape volume set. A <volume_set-group> must begin with a Volume statement, contain 1 or more <file-group>s, and terminate with an End statement. In addition, a <volume_set-group> may optionaly contain 1 or more <global-statement>s.

The Volume Statement

```
     Volume: <volid> [ ["-comment text"]
                       [,<volid>]
                       [,<volid> -comment text] ] ...;
```

The control file Volume statement specifies the tape volume set to be used in file transfer. Although the Volume statement allows multiple volume ids to be specified, only tape_ansi_ and tape_ibm_ have automatic volume switching capability and can make use of this multiple specificiation. For ntape_ and tape_mult_, if more than one volume is specified per <volume_set-group>, the first volume is used by the IO module and the remainder will be ignored. The simplest and most typical control file Volume statement would be

```
Volume: <volid>;
```

This statement causes a tape volume whose volume serial number is <volid> to be mounted on a 9-track drive. <volid> must consist of from 1 to 6 ASCII characters. If <volid> is less than 6 characters, it will be padded on the right with blanks to a length of six. If <volid> contains any of the following characters, <volid> must be enclosed in quote characters ("):

1) any ASCII control character - (tcl ignored breaks)
2) : ; , or blank - (tcl breaks)
3) the sequence /* or */ - (tcl comment delimiters)

If <volid> itself contains a quote character, the quote must be doubled and the entire <volid> string enclosed in quotes. If the first character of a <volid> is a hyphen ("-"), the <volid> must be preceded by the keyword -volume and the whole phrase must be enclosed in quotes, as seen in the Volume statement below. If the -volume keyword is omitted for such a <volid>, an error is indicated.

```
Volume: "-volume -01032";
```

A more complicated control file Volume statement might contain a volume set specification where more than one <volid> was given. The multiple <volid>s of a volume set are separated from one another by commas and are listed either in the order in which they became members of the volume set or in the order in which they are candidates for volume set membership. The entire volume set membership need not be specified in a Volume statement referencing a volume set, but the first (possibly only) member must be mentioned. Up to 64 <volid>s may be specified in a single control file Volume statement.

If it is necessary for the user to have a message displayed on the operator's console, the comment phrase can be included in the Volume statement. The comment phrase consists of the keyword -comment followed by the text of the message; this phrase in enclosed in quotes. Whenever the volume with the <volid> immediately preceding the comment phrase is to be mounted, the specified message will be displayed on the operator's console. The message may be concerned with any subject, but it is typically used to display the slot identifier of the tape being mounted when it differs from the volume label. The message, **text**, may be from 1 to 64 characters and must be a contiguous string with no embedded spaces.

Examples

```
Volume: 001234;                        mounts volume 001234
Volume: XJ56;                          mounts volume XJ56bb
Volume: "as"";56";                     mounts volume as";56
Volume: "-volume -00451";              mounts volume -00451

Volume: 070064 "-comment in_slot_1000", 070065;

                     mounts the first member of the volume set,
                     070064, displaying the message  "in_slot_1000"
                     on the operator's console. Later, volume
                     070065 may be mounted with no message
                     appearing.
```

Under certain circumstances when a decision is to be made or certain additional information is required such as for volume initialization or file expiration, the tape_ansi_ and tape_ibm_ IO modules will query the user by asking questions on the stream user/io. Refer to the MPM documentation on tape_ansi_ and tape_ibm_, section on Queries.

<volume-group> Defaults

Associated with a <volume_set-group> are a set of default characteristics. In the absence of overriding <global-statement>s or <local-statement>s, these defaults will apply to all <file-group>s within the <volume_set-group>.

1) tape IO Module used: tape_ansi_
2) density: 800 bpi
3) expiration: immediately
4) device: one
5) storage system file format: unstructured
6) mode: 9mode, ascii character code
7) tape file record format: S
8) physical block length: 8828 characters (maximum)
9) logical record length: 131091 characters (maximum)

<global-statement>

A <global-statement> changes a <volume_set-group> default. The DIM, Density, Device, No_labels and DOS <global-statement>s may appear only once in a <volume_set-group>. The Storage, Expiration, Mode, Format, Block and Record <global-statement>s may appear any number of times within a <volume_set-group>. The DIM, Density, and Device <global-statement>s are meaningful when used in conjunction with all IO modules. The No_labels and DOS <global-statement>s are meaningful only when used in conjunction with the tape_ibm_ IO module. If they are used in conjunction with any other tape IO module, an error is indicated. The Expiration, Mode, Format, Block and Record <global-statement>s are meaningful only when used in conjunction with either the tape_ansi_ or tape_ibm_ IO modules. If used with any other tape IO modules, an error is incicated. A summary of <global-statement>s and their associated IO modules follows.

Summary <global-statement>s:

DIM: <dim>;	[all 4 tape IO modules]
Density: <den>;	[all 4 tape IO modules]
Device: <number>;	[all 4 tape IO modules]
DOS;	[tape_ibm_]
No_labels;	[tape_ibm_]
Storage: <structure>;	[all tape IO modules]
Expiration: <date>;	[tape_ansi_ or tape_ibm_]
Mode: <mode>;	[tape_ansi_ or tape_ibm_]
Format: <form>;	[tape_ansi_ or tape_ibm_]
Block: <blklen>;	[tape_ansi_ or tape_ibm_]
Record: <reclen>;	[tape_ansi_ or tape_ibm_]

DIM: <dim>;

The DIM <global-statement> specifies the tape IO Module to be used in the file transfer. <dim> must be either tape_ansi_, tape_ibm_, ntape_ or tape_mult_. (See Appendix A for DIM compatibility tables.) This <global-statement> may appear only once within a <volume_set-group> or an error is indicated.

Density: <den>;

The Density <global-statement> specifies the density in which the volume is (to be) recorded. <den> must be either 800 or 1600. (bpi) This <global-statement> may appear only once within a <volume_set-group> or an error is indicated.

Device: <number>;

If a tape volume set consists of more than one volume, the Device <global-statement> can be used to control device assignment within a <volume_set-group>. This Device statement can only be used in conjunction with the tape_ansi_ or tape_ibm_ IO modules or an error is indicated. <number> specifies the maximum number of tape drives which can be used for a given <volume_set-group>. <number> is an integer in the range of 1 to 63 inclusive. The tape drives are assigned on a demand basis, and in no cases will the actual number assigned exceed the user's process device limit. If the Device statement is omitted, in any <volume_set-group>, the default maximum number of devices for the scope of the <volume_set-group> is <number = 1>; This <global-statement> may appear only once within a <volume_set-group> or an error is indicated.

No_labels;

The No_labels <global-statement> specifies that an unlabeled tape volume set is to be processed by the tape_ibm_ IO module. This <global-statement> only has meaning when used in conjunction with the tape_ibm_ IO module. If used with other tape IO modules an error is indicated. The No_labels <global-statement> is mutually exclusive with statements which depend on tape labels namely: the replace, modify and extend <local-statments>s and the DOS and Expiration <global-statement>s. If any of these appear together within the same control file <file-group>, an error is indicated.

DOS;

This <global-statement> specifies that the tapes read or written by the control file are destined for or have been produced by a DOS installation. This DOS <global-statement> only has meaning when used in conjuction with the tape_ibm_ IO module. If the DOS <global-statement> is used with any other tape IO module, an errror is indicated. As DOS tapes do not record file structure attributes in the file labels, it is necessary to specify all file attributes for each <file-group> processed with the DOS <global-statement> or an error is indicated.

Storage: <structure>;

The Storage <global-statement> specifies the internal (logical) structure of the storage system file(s) to be referenced by subsequent <file-groups>. The unstructured file is referenced as a series of 9-bit bytes; the sequential file is referenced as a sequence of records, each record being a string of 9-bit bytes. <structure> must be either "unstructured" or "sequential". This <global-statement> is to be used with any of the four tape IO modules.

Expiration: <date>;

The Expiration <global-statement> specifies the expiration date of a file to be written (created or generated). This <global-statement> only has meaning when used in conjuction with the tape_ansi_ or tape_ibm_ IO modules. An error is indicated if it is used in conjunction with any other tape IO modules. If used with other tape IO modules an error is indicated. <date> must be a contiguous string, with no embedded spaces and must be of a form acceptable to the convert_date_to_binary subroutine (see MPM writeup on convert_date_to_binary). Because overwriting a file on a tape logically truncates the file set at the point of overwriting, the expiration date of a file must be earlier than or equal to the expiration date of the previous file (if any) on the tape; otherwise an error is indicated. If an attempt is made to overwrite an unexpired file, the user will be queried for explicit permission at the time of writing. (See MPM write-up on tape_ansi_, section on Queries).

Mode: <mode>;

The Mode <global-statement> specifies the tape mode and character code to be used with subsequent <file-group>s. <mode> must be either ascii, ebcdic, or binary. This statment only has meaning when in conjuction with tape_ansi_ or tape_ibm_ IO modules. An error is indicated if this statement is used in conjucntion with any other IO modules.

Format: <form>;

The Format <global-statement> specifies the tape record format to be used with subsequent <file-group>s. <form> must be either U, F, FB, D, DB, S, V, VB, or VBS. This statement only has meaning when in conjucntion with the tape_ansi_ or tape_ibm_

IO modules. An error is indicated if this statement is used in conjucntion with any other IO modules.

Block: <blklen>;

The Block <global-statement> specifies the tape file (maximum) physical block length, in characters, to be used with subsequent <file-group>s. <blklen> must be a decimal integer, such that 18 ≤ <blklen> ≤ 6528. This statement only has meaning when usedin conjucntion with the tape_ansi_ or tape_ibm_ IO An error is indicated if this statement is used in conjucntion with any other IO modules.

Record: <reclen>;

The Record <global-statement> specifies the tape file (maximum) logical record length, in characters, to be used with subsequent <file-group>s. <reclen> must be a decimal integer, such that 1 ≤ <reclen> ≤ 131071. This statement only has meaning when used in conjucntion with the tape_ansi_ or tape_ibm_ IO modules. An error is indicated if this statement is used in conjucntion with any other IO modules.

The End Statement

An End statement must appear as the last statement of a <volume_set-group>.

End;

<file-group>

Every <volume_set-group> must contain 1 or more <file-group>s. Each <file-group> defines 1 tape/storage-system file transfer. A <file-group> must begin with a File statement, and contain a path statement. In addition, it may contain 1 or more <local-statement>s. A <file-group> is terminated by a <global-statement>, an End statement, or a File statement (new <file-group>).

The File Statement

File: <fileid>;

The File statement specifies that a new tape file is to be read or written. The tape file is identified by <fileid>. When using the tape_ansi_ or tape_ibm_ IO modules, <fileid> is the name of the tape file to be processed. When using tape_mult_ or ntape_ to process the tape, which do not make position references by "files", <fileid> is to be "*" or an error is incicated. <fileid>, the tape file identifier, is from 1 to 17 characters inclusive.

Examples

File: File1;

File: *;

See the various IO module documentation in the MPM for particular restrictions on file identifiers.

The Path Statement

path: <pathname>;

Every <file-group> must contain 1 path statement. The path statement specifies the path name of the storage system file to be read or written. <pathname> may be either a relative or absolute path name.

<local-statement>

A <file-group> may contain 1 or more <local-statement>s. A <local-statement> overrides the <volume_set-group> defaults in effect at the time a <file-group> is evaluated. A <local-statement> has no effect outside of the <file-group> in which it occurred, and may appear anywhere within the <file-group>.

These <local-statement>s operate exactly as do their <global-statement> counterparts, except that they affect only the <file-group> in which they are contained. Some <local-statement>s only have meaning when used in conjurction with a particular IO module. A summary of <local-statement>s and their associated IO modules follow.

Summary of <local-statement>s:

no_labels;	[tape_ibm_]
storage: <structure>;	[all 4 tape IO modules]
expiration: <date>;	[tape_ansi_ or tape_ibm_]
mode: <mode>;	[tape_ansi_ or tape_ibm_]
format: <form>;	[tape_ansi_ or tape_ibm_]
block: <blklen>;	[tape_ansi_ or tape_ibm_]
record: <reclen>;	[tape_ansi_ or tape_ibm_]

Additional <local-statement>s which have no global counterparts follow. Again, some <local-statement>s are for use with specific IO modules only.

number: <number>;	[tape_ansi_ or tape_ibm_]
replace;	[tape_ansi_ or tape_ibm_]
extend;	[tape_ansi_ or tape_ibm_ or ntape_]
modify;	[tape_ansi_ or tape_ibm_]
generate;	[tape_ansi_]

number: <number>;

The number statement further identifies the tape file to be used in file transfer. Only when used in conjunction with the tape_ansi_ or tape_ibm_ IO modules does the number statement have meaning. When using the tape_ibm_ IO module to write unlabeled tapes, the number statement must be specified or an error is indicated. <number> is the file sequence number; it specifies the position of the tape file within the file set. <number> must be an integer between 1 and 9999 inclusive. If the control file is to be used with the tape_in command, <number> specified in a number statement, must refer to a file within the file set and both the <fileid> specified in the File statement and the <number> specified in the number statement must refer to the same tape file; otherwise an error is indicated. When the control file is to be used with the tape_out command, <number> specifies the file to be created or replaced. When using the tape_ansi_ or the tape_ibm_ IO modules with the tape_out command, the number statement (where <number = 1>) must be explicitly included in the <file-group> to create the first file on an entirely new file set. These IO modules initialize tape volumes with a dummy file. If <number= 1> is not specified, in such a <file-group>, the first file that is written will be appended to the file set containing the initial dummy file making with the tape file sequence number 2 rather than 1.

There are two modes in which a user can set up a control file <file-group> to write a storage system file onto a tape volume. The default output mode for all IO modules causes a new file to be created on the tape. The tape_ansi_ and tape_ibm_ IO modules offer three specialized output options in addition to create mode, and the ntape_ IO module offers one output option in addition to create mode; tape_mult_ only offers create mode.

replace: <fileid>;

If an existing tape file is to be replaces (using tape_ansi_ or tape_ibm_), and its name is known, the file to be overwritten is identified by <fileid> in the replace <local-statement> and the new file to be written is identified in the File statement by <fileid>. If the file identified in the replace statment does not exist, an error is indicated.

```
File: <fileid>;
replace: <fileid>;
```

The three additional file output options available to use when writing thru tape_ansi_ or tape_ibm_ IO modules: modify, generate and extend. These three <local-statement>s do not cause the tape file labels to be recomposed, so any file attributes specified in the control file <file-group> or <volume_set-group> do not match those specified in the tape lables, will cause an error to be indicated.

extend;

The extend statement allows new data records to be appended to an existing file on an ANSI or IBM standard labeled tape without in any way altering the previous contents of the tape file. The extend statement when used with ntape_, will cause data to be appended to whatever file the tape is positioned to at the time of attaching with extend mode. The tape file to be extended is identified by the File statement or by the File statement and number statement in combination with the tape_ansi_ or tape_ibm_ IO modules. If the tape file to be extended does not exist on the tape, an error is indicated. Recorded in the labels of an ANSI or IBM labeled tape file is the version number. Initally it is 0 when the file is created. Every time a file is extended, its version number is incremented. The version number field is two digits and is reset to 0 when the 100th revision is made. ntape_ which does not write tape labels, does not deal with a version number when extending a file.

modify;

The modify statement in a control file <file-group> to be used with tape_out causes the entire contents of a file on an ANSI or IBM labeled tape to be replaced while retaining the structure of the file itself. The file to be modified is identified by the File statement, or by a combination of the File statement and the number statement.

generate;

This statement is for use with the tape_ansi_ IO module only. If used in conjuction with another tape IO module an error is indicated. Producing a new generation of a file is essentially the same as creating a new file in place of an old file, except the file identifier, the file sequence number and the file structure attributes are maintained. Recorded in the file labels of ANSI or IBM standard labeled tapes is a file generation number. This generation number is set to zero when the file is created. The process of modifying or extending does not alter the generation number; only the process of generation changes the generation number and resets the version number to zero. If the file to be generated does not exist, an error is indicated. The generation number has a field of four digits; when the generation of a file it to be incremented from 9999 it it is reset to zero.

Execution

When the control file is being executed in response to the tape_in command, the volume set named in each <volume_set-group> of the control file is mounted in turn without write rings. If any file output options appear in a control file being executed in response to the tape_in command, namely the statements create, modify, extend or generate, these statements will be ignored. When the control file is being executed in response to the tape_out command, the volume set named in each <volume_set-group> of the control file is mounted in turn with write rings. If no tape file output options are specified in a <file-group> of a control file being executed by the tape_out command (i.e. none of the statements: create, modify, generate or extend are included), the create statement will be assumed.

Comments

Comments may be inserted anywhere within the tcl program by surrounding the comment text with the comment delimiters. /* is the delimiter which begins a comment, and */ is the delimiter which terminates a comment.

Notes

File transfer is performed as described below. See Appendix A for the value of nelem associated with each tape record format.

1) tape_out: an attempt is made to read nelem characters from the storage system file to be written onto tape. For unstructured files, nelem characters is up to and including the first new_line character encountered; for sequential files, nelem characters is one logical record. The characters actually read are then written to the tape file as 1 logical record.

2) tape_in: one logical record is read from the tape file, and as many characters as were read are written into the storage system file either as a string in unstructured format of as one logical record in sequential format.

Under certain circumstances, tape records being written must be padded in accordance with a set of per-format padding rules.

(See the MPM write-ups of tape_ansi_ and tape_ibm_.) Because of the complex interrelationship between padding rules and the treatment of new_line characters when writing tape, it is reccommenaed that the following suggestions be heeded:

1) do not use F or FB format.

2) to write character data with tape_ansi_, use U, D, DB, or S format, with the maximum block length, and the record length chosen so that nelem is greater than the longest line in the storage system file.

3) to write binary data with tape_ansi_, use U, D, DB, or S format, with the maximum permissible block and/or record lengths.

4) to write character data with tape_ibm_, use VBS format with the maximum block length, and the record length chosen so that nelem is greater than the longest line in the storage system file.

5) when transfering sequential format files to tape, use a variable length record format (U, D, DB, or S) to avoid unwanted padding characters being inserted into records.

Example

Below is an example of a typical tape_out control file. The user wishes to produce 2 tapes, one for Multics, the other for an OS installation. The Multics tape will contain the source code of user subsystem SUBSYS, as well as it's object code. The OS tape will contain only the source code. The numbers at the left-hand side of the page do not actually appear in the control file, but are included only for reference.

```
                command line:     tape_out -fc -ret

    1         Volume: 001234 "-comment in_slot_000064";
    2         /* Dump source in S format */
    3         File: FILE_1;
    4         number: 1;
```

```
   6          path: SUBSYS.pl1;
   7          File:  FILE_2;
   8          number: 2;
   9          mode: binary;
  10          path: <object>SUBSYS;
  11          format: U;
  12          End;
  13          Volume: DFG054;
  14          /* append source to tape */
  15          DIM: tape_ibm_;
  16          File:  TEST_SAVE;
  17          replace: FILE_9;
  18          Expires: 2weeks;
  19          Format: VBS;
  20          Block: 4096;
  21          Mode: ebcdic;
  22          path: SUBSYS.pl1;
  23          End;
```

1)   causes volume 001234 to be mounted with the message "in_slot_000064" appearing on the operator's console. The volume defaults are set to tape_ansi_, 800 bpi, ascii, S format, block length = 6528, and record length = 131071.

2)   is a comment
The default storage system file format will be set to transfer unstructured files.

3)   causes the tape to be positioned so that the first file that will be written on tape will be FILE_1.

4)   This is required for writing the first file on a new ANSI or IBM file set at location one on the tape.

5)   specifies the path name of the storage system file to be written to tape. As the <file-group> contains no <local-statement>s, other that the number statement, the file will be written according to the current volume defaults. The tape file will be created as a new file on the volume set.

6)   causes the tape to be positioned so that the file to be written will be FILE_2.

Because the -retain option was included in the command line, the volume will not be rewound and taken down after the first file was written detached.

7) the file to be written named FILE_2 will be the second file on the tape.

8) The file to be written on tape will be a new file.

9) specifies that the file is to be written in binary mode.

10) specifies the path name of the storage system file to be written to tape.

11) specifies that the file is to be written in U format. Note that the block length will be the current volume default block length (6528), and that the record length is not applicable to U format.

12) signifies end of <volume_set-group>. The IO switch is closed and detached. The volume set is taken down and the device is released.

13) causes volume DFG054 to be mounted. The volume defaults are set to tape_ansi_, 800 bpi, ascii, S format, block length = 6528, and record length = 131071.

14) is a comment
Storage format is still unstructured.

15) changes the default volume DIM to tape_ibm_.

16) specifies name of file to be written onto tape.

17) specifies the name of the file to be overwritten. The -force option in the command line insures that the file will be overwritten even if it has not expired.

18) This new file will expire in two weeks. Until that time, it will be protected from accidental overwritting.

19) changes the default record format to VBS.

20) changes the default block length to 4096.

21) changes the default mode to ebcdic.

22) specifies the path name of the storage system file to be written. Note that the tape file will be written using tape_ibm_, density = 800 bpi, VBS format, block length = 4096, record length = 131071, and mode = ebcdic.

23) causes the volume set to be rewound and taken down as no more <file-group>s in the control file reference the current tape volume set.

Error Diagnostics

The error messages which are issued during tape_in, tape_out compilation are graded and have the form shown below.

prefix error_number, SEVERITY *severity* IN STATEMENT *m* OF LINE *n*
text of error message
SOURCE:
source statement in error

where *n* is the line number on which the described statement began and *m* is a number identifying which statement in line *n* was in error. If line *n* contains only one statement then "STATEMENT *m* OF" is omitted from the error message.

The severity numbers produce one of the following prefixes:

severity	prefix	explanation
0	COMMENT	the error message is a comment.
1	WARNING	the error message warns that a possible error has been detected. However, the translation will still proceed.
2	ERROR	the error message warns that a probable error has been detected. However, the error is non-fatal, and the translation will still proceed.
3	FATAL ERROR	the error message warns that a fatal error has been detected. Processing of the input will still continue to diagnose further errors, but no translation will be performed.
4	TRANSLATOR ERROR	the error message warns that an error has been detected in the operation of the translator. No translation will be performed.

For errors occurring during IO processing, see the IO module documentation in the MPM.

APPENDIX A

DIM Compatibility and nelem Tables

tape_ansi_

modes: ascii, binary, ebcdic
(binary and F format are incompatible)
block length (blk): 18 ≤ blk ≤ 6528 bytes

tape_ibm_

modes: ascii, ebcdic
block length: 18 ≤ blk ≤ 6528 bytes
when writing, mod (blk, 4) must = 0

tape_mult_

mode: binary (default)
block length: 8 words header, 8 words trailer,
1024 words data or 1040 words total (default).

ntape_

mode: binary (default)
no blocking -- lrecl = blk

Format	Record Length (rec)	nelem
U	N/A	= blk
F	= blk	= rec
FB	mod (blk, rec) = 0	= rec
D	= blk	= rec
DB	≤ blk	= rec
S	1 ≤ rec ≤ 131071	= rec
V	= (blk - 4)	= (rec - 4)
VB	≤ (blk - 4)	= (rec - 4)
VBS	1 ≤ rec ≤ 131071	= rec

(END)