To: Distribution

From: J. A. Weeldreyer

Date: February 28, 1975

Subject: Multics Removable Disk I/O Module

INTRODUCTION

There appears to be substantial technical justification for the development of a capability on Multics to perform explicit input/output operations from/to disk, treating it as a removable medium. It would be desirable that this capability be available via language I/O statements, rather than merely through subroutine calls. This facility could be used to provide for disk file transfers between Multics and other (Multics or non-Multics) computer systems. Also, the backup-to-disk application needs a facility such as this. Finally, this capability will be useful to other system processes,as well as user processes, which require explicit disk I/O.

Suggestions and comments will be greatly appreciated. Please send them to me by Multics mail on System M (Weeldreyer Multics), or give me a call at (HVN) 357-3182.

GENERAL DISCUSSION

The following possibilities were considered prior to deciding which capabilities should be implemented in the initial version of the Removable Disk I/O Module:

1. Support of PL/I stream I/O,
2. Support of PL/I record I/O,
3. System controlled sharing of disk by multiple users,
4. Support of a basic, physical device oriented I/O capability which would be useable via PL/I record I/O statements,
5. Support of stranger pack formats, e.g. GCOS removable packs,
6. System controlled label checking as a form of disk pack access control, and
7. Support of some form of device accounting.

The following assumptions affected the decision:

1. A new storage system with a removable segment facility is to be implemented for MR 4.0. Therefore, most applications requiring disk for Multics files will (or should) utilize this feature with vfile_.

2. There may be some system functions, e.g. backup, which will find the Removable Disk I/O Module useful.

3. Another application for this capability would be to provide access to stranger, e.g. GCOS, disk packs.

Under these assumptions, it becomes rather obvious that there is no immediate need to support the full PL/I I/O capability since this will soon be available on another form of removable disk via the new storage system and vfile_. Also, the capability for system-controlled sharing of removable disk files will soon be provided by the new storage system.

Clearly, some form of "negative" label checking should be implemented prior to the advent of the new storage system. This would prevent the mounting of and tampering with removable file system packs. However, this would logically be a function of the Resource Control Package (rcp_) rather than this I/O module. The only other relatively near-term possible requirement for label checking would be for the backup-to-disk application. However, backup is a trusted system process running in ring 1, and will perform its own label checking as part of a comprehensive data validation scheme. Therefore, it does not appear necessary to provide a label checking capability in the initial version of this I/O module, although it may be prudent to provide for the reservation of a label area on each pack to prevent the necessity for user program modification if label checking is eventually implemented.

The device accounting function is something which is not unique to any particular I/O module, and would therefore appear to be a function of rcp_.

Although we have assumed that one possible application of a removable disk I/O module would be the accessing of stranger packs, it is not clear at this point that there is a great customer demand for such a facility, nor that it would be desirable to define stranger formats to be supported without some customer feedback. Therefore, the logical approach would seem to be the providing of a very basic initial capability which would provide a means of accessing the data on a disk pack, but would leave the formatting/unformatting of the data as an exercise for the ambitious user programmer. If, at some time in the future, it becomes apparent that a more comprehensive stranger pack support capability is required, such a facility (or facilities)

could be developed as "piggy-back" I/O modules which would utilize this "basic" I/O module.

This approach also satisfies the other near-term requirements for a removable disk I/O module. The backup_to_disk facility requires only a means of accessing the disk, since it will already be in existance when the Removable Disk I/O Module comes into being, and will already have the capability to format/unformat its data. Finally, such a facility would provide for language I/O from/to disk because it will be accessable via a limited subset of the record I/O statements of PL/I and FORTRAN.

PROPOSED I/O MODULE DESCRIPTION

I/O MODULE

Name: rdisk_

This I/O module supports I/O from/to removable disk packs. Only direct mode files are supported.

Entries in this module are not called directly by users; rather, the module is accessed through the I/O system. See the MPM section, the Multics I/O System, for a general description of the I/O system, and see the MPM section, File I/O, for a discussion of files.

Attach Description

The attach description has the following form:

rdisk_ device_id pack_id -opt1- -optn-

1. device_id — is a character string identifying the model number of the required disk device. Currently, only the DSS191 is supported. The device_id for the DSS191 is "D191".
2. pack_id — is a character string identifying the disk pack to be mounted.
3. opti — may be one of the following options. An option may occur only once.

   -write — indicates that the disk pack is to be written. If omitted, the operator will be instructed to mount the pack write inhibited.

-size n	indicates that the value of n is to override the value of buff_len as a record size limit for the read_record operation. (See notes.)

The attachment causes the specified disk pack to be mounted on a drive of the specified type. The privileged version of the disk attach mechanism will be automatically invoked for system processes. (See the I/O Interfacer documentation.)

Opening

The only opening modes supported are direct_input and direct_update. If an I/O switch attached via rdisk_ is to be opened for update, the -write option must occur in the attach description. This operation has no effect on the physical device.

Delete Record Operation

This operation is not supported.

Read Length Operation

This operation is not supported.

Read Record Operation

If the amount of data to be read does not terminate on a sector boundary, the excess portion of the last sector will be discarded. A zero code will be returned in this case. (See notes for further discussion.)

Rewrite Record Operation

This operation is the only output operation supported. If the amount of data to be written does not terminate on a sector boundary, the remaining portion of the last sector will be filled with binary zeros. A zero code will be returned in this case. (See notes for further discussion.)

Seek Key Operation

This operation will return a zero status code for any key which is a valid sector number. The record length returned will always be 64 (current physical sector size) for any valid key. (See notes for further discussion.)

Control Operation

The following orders are supported when the I/O switch is open, except for getbounds, which is supported while the switch

is attached.

changepack — causes the current pack to be dismounted and another pack to be mounted in its place. The info_ptr should point to a varying character string (maximum of 32 characters) containing the identifier of the pack to be mounted.

getbounds — causes the lowest and highest sector numbers accessible by the caller under the current modes to be returned. The info_ptr should point to a structure like the following:

```
dcl 1 bounds,
      2 low fixed bin (35),
      2 high fixed bin (35);
```

setsize — causes the value of the record size override setting to be reset. The info_ptr should point to a fixed bin(35) quantity containing the new override value.

Modes Operation

The modes operation is supported when the I/O switch is attached. The recognized modes are listed below. Each mode has a complement indicated by the character "^" (e.g. "^label") that turns the mode off.

label,^label — specifies that a system-defined number of sectors at the beginning of the pack are reserved for a pack label, and that a seek_key operation is to treat any key within this area as an invalid key. (Default is on.)

alttrk,^alttrk — specifies that the pack has been formatted with the assignment of alternate tracks, so that a system-defined number of sectors at the end of the pack are reserved for an alternate track area. Therefore, a seek_key operation is to treat any key within that area as an invalid key. (Default is off.)

wrtcmp,^wrtcmp	specifies that the Write-and-Compare instruction,rather than the Write instruction will be used for the rewrite_record operation. This causes all data written out to be read back in and compared to the data as it was prior to being written. This mode should be used with discretion, since it doubles the data transfer time of every write. (Default is off.)

Write Record Operation

This operation is not supported.

Closing

The closing has no effect on the physical device.

Detaching

The detachment causes the disk pack to be dismounted.

Notes

This I/O module provides a very elementary, physical device oriented I/O facility, and is the basic user-level interface to a disk device. All operations are performed through calls to various I/O Interfacer (ioi_) and Resource Control Package (rcp_) entries. This I/O module provides the capability to read/write a caller-specified number of characters to/from a disk_pack, beginning at a caller-specified sector number. Currently, the DSS191 is the only device type which is supported.

The entire disk pack is treated as a keyed direct file, with keys interpreted literally as physical sector numbers. Hence, the only allowable keys are those which can be converted into a fixed binary integer which falls within the range of valid sector numbers for the given disk device under the current modes, as returned by the getbounds control order.

If an attempt is made to read or write beyond the end of the user-accessible area on disk, the code error_table_$device_end is returned. If a defective track is encountered or if any other unrecoverable data transmission error is encountered, the code error_table_$device_parity is returned.

The record length is specified via the buff_len parameter in the read_record operation, and via the rec_len parameter for the rewrite operation, unless overridden by a -size option in the attach description. (Since the file is defined to consist of the

entire pack, the write operation has no meaning in this I/O module.)

The following items must be considered when using this I/O module with language I/O:

1. Device Attachment and File Opening:

   a. PL/I: A file can be attached to a disk pack in PL/I by specifying the appropriate attach description in the title option of an open statement. The open statement should also specify the record and keyed attributes, plus either the input or update attribute, as is appropriate. After opening, the desired modes should be set, and the current sector bounds should be obtained, through direct calls to iox_$find_iocb, iox_$modes, and iox_$control.

   b. FORTRAN: It is not possible to attach a file to a disk pack within FORTRAN. Here, the attachment must be made external to the FORTRAN program, e.g. via the io_call command or through use of a PL/I subroutine. FORTRAN will automatically open the file with the appropriate attributes. Also, it is impossible to set modes or obtain sector bounds from within FORTRAN. This should be done through use of a PL/I subroutine prior to the first FORTRAN reference to the file.

2. Input:

   a. PL/I: The PL/I read statement with the into and key options is used to read data from a disk pack. The input record length (buff_len) is determined by the size of the variable specified in the into option. The set option should not be used. The key should be a character string containing the character representation of the desired sector number.

   b. FORTRAN: The unformatted, keyed version of the FORTRAN read statement is used. The key must be an integer, the value of which is the desired sector number. In FORTRAN, buff_len has no relationship to input variable size. Hence, the -size option must be specified in the attach description if the disk pack is to be read via FORTRAN. The size should be set to the length of the longest expected record.

3. Output:

   a. PL/I: The PL/I rewrite statement must be used to perform output operations to a disk pack. The from and key options must be specified. The size of the variable referenced in the from option determines the

length of the record written to disk. The key should be a character string containing the character representation of the desired sector number.

b. FORTRAN: The unformatted, keyed version of the FORTRAN write statement must be used to perform output operations to a disk pack. The size of the output record is determined by the amount of data specified in the write list. The key must be an integer, the value of which is the desired sector number.