```
/midas/local/milano.mods, attached here..pl60
.mb 3
.po 0
.he$ELISM IFCTR Software note - Report on tests of VMS-Unix file
exchange$ELITE
.fo$ELISM VMS-Unix file exchange - 22 Aug 1990 - Page #$ELITE$FF
```

# A report on tests of VMS-Unix file exchange

prepared by L.Chiappetti - IFCTR
22 Aug 1990

## 1. Purpose

I have recently performed a number of tests of different modalities of file exchange between the VAX (VMS) and Sun (Unix) systems at IFCTR, using different products and arrangements. The purpose of such tests was manifold :

to assess the performance and ease of use of different products (standard Digital's Decnet, Digital UCX 1.0 VMS to Ultrix Connection, standard SunOS TCP/IP, Sun's Sunlink/DNI Decnet emulation) and different ways of file exchange (remote file access with Decnet, ftp file transfer, shared file access with NFS), also in order to make a choice of the products to keep permanently installed.

in particular to assess the performance and ease of use of such products and modalities for the transfer of data as used by astronomical packages (like IRAF and MIDAS) which are or will be installed on the above mentioned machines.
finally, to assess the performance and ease of use of such products and modalities in the case the files have to be accessed by Fortran programs (after a file transfer, or over the network), also in consideration of the possible future development of new software for X-ray astronomy, which is aimed to portability. This is made in the framework of an ideal scheme, which is described in section 2 below. The assessment shall be made jointly with the evaluation of the peculiar i/o needs of such new software.

## 2. Considerations

The results of the tests are summarised in a series of tables below. More material (in the form of informal notes taken in real time during the tests) is available on request (in print or in file form) on the following topics (all files reside on /home/lucio/connect on Sun):

sunvax.lis ("Connecting a Sun and a Vax System. Comparison between VMS-Ultrix-Connection (UCX) and Sunlink DNI")

vaxascii.form ("Layout of standard text files on Vax and Sun")

vaxbinary.form ("Layout of binary files on Vax and Sun")

astrobinary.form ("Sun-Vax Exchange of binary files. Additional notes on common astronomical formats")

.cp3

The tests have been as complete as possible consistently with the current installation. In particular more tests would be useful in the following contexts :

Testing file exchange with other types of Unix workstations (e.g. HP 9000, DECStation)

Testing the exchange of MIDAS-format files (this has not been done, as MIDAS is not yet installed on the VAX) and IRAF STF-format files (this has not been done as IRAF 2.8 on Sun has a bug which disallows support to such files) to complete the overview of common astronomical formats (only IRAF OIF and FITS have been tested so far).

The following guidelines have been followed during the test : it was not felt particularly important to assess the speed of the data transfer, but instead to : verify the integrity of the data transfer (verify the files were usable at the destination immediately, or with minimal reformatting); verify the limitations of the transfer modalities (e.g. imposed by system security features, like file protections etc.); verify the ease of use of each modality (where single, legible commands are preferred over awkward syntax strings, or sequences of more commands); verify the usability of the transferred files by normal Fortran programs; verify the accessibility of remote files by normal Fortran programs.

The following ideal model was considered (in view of future software developments) :

There are essentially two types of files: text and binary. Text files are used for small quantities of data, which shall be manipulated interactively (edited, printed, typed). These files can have a variable record length.

Binary files are used for larger quantities of data. In this case it is convenient to use unformatted data transfer for speed, and direct access modality for generality of use (in the case only part of the content of the file has to be manipulated, and if it has to be manipulated in-place, without copying to another file).

These two cases are offered within any standard Fortran compiler, and are considered the two main modes of file access. All other modes offered by such compilers (the other standard combinations, like formatted direct access, or sequential unformatted; as well as all the non-standard forms offered by some compilers, keyed, indexed etc.) are disregarded.

This model offers a great simplicity, and should easily be implementable on any single system. One of the aims of this work, was to verify the applicability of such model also to the case where the data may reside on different machines of an heterogeneous network.

.cp5

## 3. Conclusions

From the point of view of the <u>evaluation of the products</u> for their usage one may conclude what follows :

DNI does not offer any advantage for Sun-based file transfer (it is slower and has a nasty syntax). However it is required to perform remote login from/to Sun to/from Decnet nodes (and in this case the associated te100tool is quite good), and also to access Sun files via DECNET from VAX. The latter capabilities are not supplied by UCX 1.0.

UCX 1.0 has a number of limitations (has no telnet or remote login, has no way to handle mail, has only a limited ftp emulation), which may make preferable a different TCP/IP emulator, if one could be found at a comparable price. Anyhow :

The usage of NFS on the VAX is the nicest way to access the VAX files from Sun, however it is unable to handle correctly the standard VAX text files (i.e. those not in stream mode).

The usage of FTP on either machine seems the safest way to handle file exchange under the majority of conditions (including a correct handling of VAX text files), although it is less comfortable to use than NFS.

From the point of view of the exchange of data in prepackaged astronomical formats, one may issue the simple recommendation : for exchange of image files within IRAF, use the FITS format only.

From the point of view of generic exchange of files, one may issue the following recommendations :

For applications which require exchange of text files to be used immediately on both machines (or accessed remotely by programs), use (on VAX) STREAM_LF organization, carriagecontrol=LIST. No action is required on Sun. The above organization can be easily forced within a Fortran program. However it is not clear how this can be specified in the case of a file to be created or manipulated with the editor.

For integrity of the transfer of binary data, always remember to specify the binary (or "image") transfer mode.

Note there are problems with the exchange of binary files, which are not always solved easily in a bilateral way.

Finally, from the point of view of the development of a complete system for X-ray astronomy, suitable for VMS and Unix , one is led to the following considerations:

One should first decide which one of the following alternatives is the most suitable for an analysis environment :

A portable software (in source form), which runs in an identical form on completely separate machines. There is no data exchange among the machines (except for raw data, or data in "independent" formats like FITS). Note that this is a perfectly acceptable solution, and corresponds e.g. with the baseline described in the XAS document.

A portable software, where however it is foreseen some form of non-sporadic data exchange among the different machines. In this case data should be explicitly exchanged, i.e. moved from one machine to the other, passing through some intermediate format, or via some conversion utility.

A distributed software, capable of transparently accessing data residing on a different machine from within each component program.

In this respect one can make additional considerations :

If the latter (network-distributed) system is wished, the test reported here show that the "ideal model" described in section 2 shall be abandoned in favour of a different, Unix-like, or IRAF-like, approach, like the one proposed by Morini in the XAS document VOS section (and which I am still considering as not desirable, form the point of view of the elegance and legibility of the program).

Or alternatively renounce to the direct access files, and use a different form of binary files (it could be noted that not all type of data structures require necessarily direct access files : if for instance one ALWAYS reads a COMPLETE image or light curve).

However there is no compulsory need or advantage in the choice of the third solution w.r.t. one of the other two.

Moreover, even if a dedicated layer of i/o software capable of handling transparently different file organizations across the network is written, one still remains with the problem that there is no transparency, because the internal representation is anyhow different. Conversion utilities are still necessary (e.g. byte-swapping, double-byte-swapping, floating point format conversion, all that mixed according to the file structure, etc.), and are possibly too cumbersome to be done "on the fly".

Therefore one is lead at least to the second solution, the ones involving conversion utilities (which may be combined at the data transfer level). And since a conversion is needed for the internal representation, could it also not handle the different file organization ?

One may therefore conclude that, using a solution which is simpler, viable, and corresponding to the actual needs (like the first or the second one) one could still consider the "ideal model" valid, and, pending a verification of the needs of the individual data structures and of the i/o efficiency, use the two essential types of "Fortran-like" files (text and direct access binary).
.cp50

## TABLES

### A. Preliminary definitions

The following preliminary definitions apply to the summary tables presented here below (they do NOT apply to the informal notes quoted in section 2 above):

For the sake of concision it has been felt necessary to define the various communication modalities with a shorthand notation which is reported in the table below. Please note that the first column gives the shorthand notation, the second column the full name/description and the third column an example applied to file transfer (however the communication modalities are not limited to file transfer, but involve other features like directory access, etc.).

The shorthand notation generally includes the name of the machine on which the user is assumed to be acting when issuing the command (DECNET implies VAX). It further includes the name of the protocol. Syntactic variants are designed with an additional digit.

$RULE1650,5

**Table a.1: communication modalities**

$RULE1650,3

| | | |
|---|---|---|
| VAX FTP | DEC UCX FTP | using FTP VMS style on VAX[1] |
| VAX FTPU | DEC UCX FTP/Ultrix | using FTP Unix style on VAX[1] |
| SUN FTP | SunOS ftp | using ftp on Sun[2] |
| SUN NFS | SunOs NFS | using sun cp command via NFS[3] |
| SUN DNI | Sunlink DNI | using sun dnicp command[4] |
| DECNET.1 | VAX DecNet | using VAX COPY in pseudo-VMS style[4] |
| DECNET.2 | idem | using VAX COPY in foreign style[4] |

$RULE1650,3

1: implies ftp running at the other (Unix) end
2: implies UCX or other FTP emulation at the other (VMS) end
3: implies UCX or other NFS emulation at the other (VMS) end
4: implies Decnet or an emulation (like DNI) running at the other end

*  note there is no nfs access of Sun files from VAX
$RULE1650,5

Some of the commands described below require a complex syntax involving use of separators (quotes, double quotes, etc.). In order to keep the command lines short, and to evidentiate the real elements (filenames) from the separators, a shorthand notation is used. Wherever one of the syntax elements listed in the first column of Table a.2 is used, it shall be replaced by a more complex name or expression, as specified below.

.cp5
$RULE1650,5

**Table a.2: syntax elements**
$RULE1650,3

| | |
|---|---|
| vax | the name of the vax host |
| sun | the name of the sun host |
| dev | a VMS device (e.g. DUA0) |
| dir | a generic directory or subdirectory |
| fsdir | a Unix filesystem root directory (e.g. /, /usr, /home) |
| vaxdir | a VMS directory, full form is dev:[dir.dir] |
| sundir | a Unix directory, full form is /fsdir/dir/dir |
| /vax | the mount point of a VAX disk on Sun using NFS |
| fn | a VMS filename, or an Unix full file name |
| ft | a VMS filetype |
| v | a VMS version number |
| fmap | the mapping of a VMS fn.ft;v onto Unix |
| user | an user (account) name |
| pwd | a password |
| | |
| vmsfile | a file on VAX vaxdir:fn.ft;v |
| sunfile | a file on Sun sundir/fn |
| nfsfile | a file on VAX mapped thru NFS /vax/dir/fmap |
| psefile | a file on Sun seen from Vax fsdir:[dir]fn |

$RULE1650,5

## B. File transfer

The following table gives the syntax for file transfer with the various communication modalities under default conditions. Default conditions mean that no special accounts or proxy accounts are set up on either machine (except when compulsory, that is in UCX is compulsory to define an account on VAX serving a given user on Sun), no dedicated shell scripts or .COM files are in use and no logical names are used on VAX to abbreviate a pathname (or environment variables on Sun with dnicp to set-up the remote username).

The only implied abbreviations is ftpu, which is a symbol defined on vax as ftpu == "$UCX$FTP/ULTRIX" to access FTP in Unix notation (UCX only).

As a complete file transfer may imply the issuing of many commands and subcommands (e.g. using ftp one shall first "log-in" into ftp, then copy, then exiting ftp), a pipe (|) is used in the table to separate commands which have to be typed separately (each one terminated with its own carriage return : no actual pipe shall be used !!!).

The rating column is a personal evaluation of the user-friendliness of the command.

.cp5
$RULE1800,5

**Table b.1: syntax for file transfer**

$RULE1800,3
from VAX to Sun
$RULE1800,3$ELISM

| Modality | on host | command(s) to be given | Rating |
|---|---|---|---|
| VAX FTP | vax | FTP \|connect sun\|user\|pwd\|put vmsfile "sunfile"[1]\|...\|exit | messy |
| VAX FTPU | vax | ftpu\|open sun \|user\|pwd\|put vmsfile sunfile \|...\|quit | good |
| SUN FTP | sun | ftp \|open vax \|user\|pwd\|get vmsfile sunfile \|...\|quit | good |
| SUN NFS | sun | cp nfsfile sunfile[2,3] | ideal |
| SUN DNI | sun | dnicp -u user -p pwd 'vax::vmsfile' sunfile[3] | messy |
| DECNET.1 | vax | copy vmsfile sun"user pwd"::psefile | fair |
| DECNET.2 | vax | copy vmsfile sun"user pwd"::"sunfile" | messy |

1: doublequotes are used to inhibit conversion to upper case (note that such conversion occurs also if no sunfile name is specified)
2: or any valid alias (e.g. copy)
3: a dot shall be specified for sunfile, in order to default the output file name

$ELITE$RULE1800,3
     from Sun to Vax
$RULE1800,3$ELISM

| Modality | on host | command(s) to be given | Rating |
|---|---|---|---|
| VAX FTP | vax | FTP \|connect sun\|user\|pwd\|get "sunfile" vmsfile\|...\|exit | messy |
| VAX FTPU | vax | ftpu\|open sun \|user\|pwd\|get sunfile vmsfile \|...\|quit | good |
| SUN FTP | sun | ftp \|open vax \|user\|pwd\|put sunfile vmsfile \|...\|quit | good |
| SUN NFS | sun | cp sunfile nfsfile[1] | ideal |
| SUN DNI | sun | dnicp –u user –p pwd sunfile 'vax::vmsfile'[2] | messy |
| DECNET.1 | vax | copy sun"user pwd"::psefile vmsfile[2] | fair |
| DECNET.2 | vax | copy sun"user pwd"::"sunfile" vmsfile[3] | messy |

1: a dot shall be specified for nfsfile, in order to default the output file name
2: a vaxdir (also []) shall be specified to default the output file name
3: it is not possible to specify a vaxdir; if a vaxfile is specified all input files are concatenated there (see below ???)

$ELITE$RULE1800,5

When a proxy account, or default account or logical name or other mechanism is used the above syntax may get somehow simplified.

.cp5
$RULE1800,5

**Table b.1/bis: simmplified syntax for file transfer**
$RULE1800,3
     from VAX to Sun
$RULE1800,3

VAX FTP   no simplification possible, there is no mechanism to define aor FTPU default account on the remote node; it is also not possible toput commands for FTP in a .COM file

SUN FTP   ftp vax |get vmsfile sunfile |quit
          the above is possible if a .netrc file is created (see ftp Unix man page), which contains the default name and password, plus other interesting features like macro definitions.  There is only one default name per remote machine.

          alternatively, it is possible to call ftp

<infile, where infile contains all necessary ftp commands (but the password shall be supplied interactively anyhow). This may be useful for long sequences of (fixed) commands

it is also possible to write a shell script, which contains the ftp commands in the form of an "here document". As the "here document" may contain variables, it is possible to pass arguments (the file names) to the script. It is always necessary to specify the password interactively.

SUN NFS   no further simplification necessary

SUN DNI   dnicp -p pwd 'vax::vmsfile' sunfile
the user can be defaulted by a setenv DNI_USER user (may getaltered)

dnicp 'vax::vmsfile' sunfile
this is possible under two conditions :

the first is that a default Decnet account is created on VAX, which handles all remote calls with unspecified user. Of course one will be limited by the access limitations attributed to the default Decnet account (a WORLD user). Moreover any created vaxfile will belong to such account.

the second one is to define a proxy account on VAX serving a specific sun user. I have been unable to test this for inconsistency in the available documentation.

DECNET.1  copy vmsfile sun::psefile
DECNET.2  copy vmsfile sun::"sunfile"

There are two ways to achieve this simplification.

The first one is to define a default DNI account on Sun. This is functionally equivalent to a default Decnet account on VAX and shares the same limitations. An additional inconvenience is that created sunfiles will belong to user dni.wheel.

The second one is to use a logical name to define a complete path of the form

```
     define lname "sun""user pwd""::fsdir:[dir.]"
```

and use lname instead of the node name. This works best for filesystem roots.

$RULE1800,3
     from Sun to VAX
$RULE1800,3

| | |
|---|---|
| VAX FTP | no simplification possible. or FTPU |
| SUN FTP | see discussion above for VAX to Sun case |
| SUN NFS | no further simplification necessary |
| SUN DNI | see discussion above for VAX to Sun case |
| DECNET | see discussion above for VAX to Sun case |

$RULE1800,5

The next tables indicates how file protections are mapped from one system to the other (consider that VMS has four protection bits for four user cathegories, while Unix has three protection bits for three user cathegories), and also how file ownership information is propagated. The first tables still refer to a default case (no proxy account set-up, unless when compulsory, i.e. UCX).

The following user classes are considered :

| | |
|---|---|
| simple | a generic sun user without UCX proxy |
| fulluser | a sun user with an UCX proxy server account on Vax |
| system | sun root with SYSTEM as UCX proxy server on Vax |
| logged | any user logged-in via ftp |
| current | the current logged-in user |
| decuser | the username specified in the Decnet access control string |

Please do not confuse the session login with the ftp login !!

.cp15
$RULE1800,5

**Table b.2: propagation of file ownership**

$RULE1800,3
     from VAX to Sun
$RULE1800,3

| Modality | Owner of transferred sunfile | Which vax files can be copied ? |
|---|---|---|
| VAX FTP | logged | as current vax user |
| VAX FTPU | logged | as current vax user |
| SUN FTP | current sun u. | depends on current sun user and on the vax user logged in via ftp as well :<br>system sunuser can access everything<br>full sunuser can access anything but system<br>simple sunuser can access nothing but the vax area of a vax user with same name<br>if a full sunuser logs-in to ftp as a simple sunuser, or viceversa, the access is limited to the ftp logged user. |
| SUN NFS | current sun u. | anything his proxy vax user is entitled to access (nothing if simple sunuser) |
| SUN DNI | current sun u. | as specified decuser |
| DECNET | decuser.wheel[1] | as current vax user |

1 : group wheel is assigned irrespective of current group-id

$RULE1800,3

from Sun to VAX

$RULE1800,3

| Modality | Owner of transferred vaxfile | Which sun files can be copied ? |
|---|---|---|
| VAX FTP | current vax u. | as logged sunuser |
| VAX FTPU | current vax u. | as logged sunuser |
| SUN FTP | logged | as current sun user |
| SUN NFS | proxy vax user | as current sun user |
| SUN DNI | decuser (-u) | as current sun user |
| DECNET | current vax u. | as specified decuser |

$ELITE$RULE1800,5

.cp15

$RULE1800,5

**Table b.3: propagation of file protection**

$RULE1800,3

from VAX to Sun

$RULE1800,3

| Protection of transf. sunfile | Limitations in accessing vaxfiles |
|---|---|
| | According to OWNER or WORLD access (GROUP and SYSTEM are ignored) |

|  |  |  |
|---|---|---|
| VAX FTP or FTPU | rw-rw-rw | only RW vaxfiles are transferred correctly: an empty sunfile is created is the vaxfile has only READ access (sic !) |
| SUN FTP | def sun umask or unchanged | Only vaxfiles with R access are transferred the other are rejected (no sunfile created) |
| SUN NFS | mapping 1 | Only vaxfiles with R access are transferred |
| SUN DNI | mapping 1 | Any vaxfile (even if R-protected) is copied |
| DECNET | rwxrwxrwx | Any vaxfile (even if R-protected) is copied |

$RULE1800,3

from Sun to VAX

$RULE1800,3

|  | Protection of transf. vaxfile | Limitations in accessing sunfiles |
|---|---|---|
|  |  | According to OWNER or WORLD access (GROUP and SYSTEM are ignored) |
| VAX FTP | set/prot default | only sunfiles with r access are copied |
| VAX FTPU | idem | idem |
| SUN FTP | idem | idem |
| SUN NFS | mapping 2 | idem |
| SUN DNI | mapping 3 | idem |
| DECNET | set/prot default | idem |

$RULE1800,3

mapping 1: VAX to Sun; acceptable

R into r, W into w, E into x, D ignored
OWNER maps into user, GROUP into group, WORLD into other, SYSTEM ignored

mapping 2: Sun to VAX; acceptable (overcautious)

r into R, w into WD, x into E
user maps into OWNER and SYSTEM, group maps into GROUP (but write protection is ignored, i.e. vaxfile will have no write access), other maps into WORLD (but write protection is also ignored)

mapping 3: Sun to VAX; dangerous
r into RD, w into WD, x into ED, none into D (sic !)
SYSTEM is given RWED access, user maps into OWNER
Note that if NO access is given for a cathegory, the corresponding cathegory on Vax will get delete access !!

$RULE1800,5

The following table indicates whether overwriting of an existing file is allowed or not (for some commands, like Unix cp, this can be enabled or disabled). The table also indicates (for VMS system which support version numbers) if a new version is created.

$RULE1800,5

**Table b.4: file overwriting**

$RULE1800,3
from VAX to Sun
$RULE1800,3

| | |
|---|---|
| VAX FTP | existing sunfile is overwritten |
| VAX FTPU | existing sunfile is overwritten |
| SUN FTP | existing sunfile is overwritten |
| SUN NFS | according to setting of Unix cp command (overwrites or asks) |
| SUN DNI | existing sunfile is overwritten |
| DECNET | exisiting sunfile is overwritten |

$RULE1800,3
from Sun to VAX
$RULE1800,3

| | |
|---|---|
| VAX FTP | a new version of the vaxfile is created |
| VAX FTPU | a new version of the vaxfile is created |
| SUN FTP | a new version of the vaxfile is created |
| SUN NFS | according to setting of Unix cp command (overwrites or asks) no new version of the vaxfile is created |
| SUN DNI | a new version of the vaxfile is created |
| DECNET | a new version of the vaxfile is created |

$RULE1800,5

Finally the following table summarizes other miscellaneous problems with the various transfer modalities (essentially concentrates to multiple file copying, and to syntactical snags).

.cp10
$RULE1800,5

**Table b.5: misc problems in file transfer**

$RULE1800,3
from VAX to Sun
$RULE1800,3

| | |
|---|---|
| VAX FTP | No transfer of multiple files into single destination It is not possible to specify the output file name of each file (but the last) in case of multiple file transfer (the other default to an upper case version of the vax name). |
| VAX FTPU | Multiple file put cause a crash; use mput instead However mput does not like wildcards, and allows only default names for output files (cannot be specified by user) |
| SUN FTP | if a vaxdir is specified, is interpreted as part of the destination unix file name.No multiple file get (gives error); also mget (silently) does not work. |
| SUN NFS | Target directory must be specified (as .) as usual with cp to copy (also multiple and wildcarded) files with same name |
| SUN DNI | It is quite slow (it opens a SERVERnnn process on VAX which remains there for some time; it also leaves a log file around); target directory etc. as SUN NFS |
| DECNET.1 | multiple file copy works; use *.* to default output file name |
| DECNET.2 | multiple file copy gives error, wildcards in output names are not allowed (you may find a file called * lying around). The sun / filesystem maps into root: pseudo-disk. |

$RULE1800,3
     from Sun to VAX
$RULE1800,3

| | |
|---|---|
| VAX FTP | no multiple gets and no wildcards |
| VAX FTPU | multiple gets of lists only, no wildcards |
| SUN FTP | mput possible; if only the output path is specified, the output file name is .;1 |
| SUN NFS | asks permission to overwrite (see tab. b.4) |
| SUN DNI | creates a new version (see tab. b.4) |
| DECNET.1 | multiple files OK; specify current vaxdir target as [] |
| DECNET.2 | concatenates multiple files into a single output file |

$RULE1800,5

The following tables deal with the format of the files, and whether this is altered by the file transfer. Two main cathegories of files are considered : text files and binary files.

The main problem here is that Unix and VMS use completely different philosophies : a simple (too simple !) one for Unix, which essentially handles only variable records terminated by newlines (linefeeds), and a complex (too complex !) one for VMS, which allows a multiplicity of file organizations and record formats and attributes.

.cp4
For simplicity, and according to the ideal scheme described in section 2, only a limited number of combinations has been tested, namely "standard text files", "binary Fortran direct access files", and a few types of astronomical formats as produced by IRAF.

> Concerning **text files** the standard format for sunfiles is the traditional Unix format (variable-record, newline-terminated); the standard format for vaxfiles is the (organization=sequential, recordformat=variable, carriagecontrol=list, max = nn bytes ) file. However a VAX is able to recognise many other formats, inclusive of the Unix format, which is seen as (organization=sequential, recordformat= STREAM_LF, cc=none, max = 512 bytes).

Please remember that the carriagecontrol attribute is an attribute of the file, and is nowhere stored within the file, nor does it affect the file content, but only the way it is displayed (a cc=list is OK, a cc=Fortran is displayed without the first character in each line, a cc=none is displayed all on one single line)

The next table gives the format for text files. All transfers are almost transparent, in the sense that the transferred file can be used on the destination machine. The main exception are vaxfiles copied to Sun via NFS. Note that the VAX Editor gives a warning "non standard text file" when accessing a STREAM_LF file, however it is perfectly able to work with it. A minor problem on the VAX is due to the fact that the "maximum record length" of a file transferred from Sun is determined in a way which depends on the transfer modality (this may give inconsistencies when trying to join files with different characteristics).

$RULE1800,5

**Table b.6: format of standard ASCII text files**

$RULE1800,3
Copying a standard text file from VAX to Sun
$RULE1800,3

| Modality | vaxfile format | resulting sunfile format |
|---|---|---|
| VAX FTP | standard text | unix |
| VAX FTPU | idem | unix |

|  |  |  |
|---|---|---|
| SUN FTP | idem | unix |
| SUN NFS | idem | file is not legible |
| SUN DNI | idem | unix |
| DECNET | idem | unix |
|  |  |  |
| Any | STREAM_LF | unix |

.cp8
$RULE1800,3
     Copying a standard text file from Sun to VAX
$RULE1800,3

| Modality | sunfile format | resulting vaxfile format |
|---|---|---|
| VAX FTP | unix | STREAM_LF cc=list  max n bytes |
| VAX FTPU | idem | idem |
| SUN FTP | idem | idem |
| SUN NFS | idem | STREAM_LF cc=none  max 32767 bytes |
| SUN DNI | idem | STREAM_LF cc=list  max undefined |
| DECNET | idem | STREAM_LF cc=list  max n bytes |

$RULE1800,5

Concerning **binary files**, the "standard" format considered here is the one produced by a Fortran program creating a direct access, unformatted file (by definition, fixed record length). The standard VMS file format in this case is (organization=sequential, recordformat=fixed, carriagecontrol=none). The "standard" Unix file format in this case is a plain stream of bytes (which is totally not standard for Unix !); in particular the information on the record length is stored nowhere in the file directory, and is therefore unknown to the system ! VAX sees such Unix files as any other Unix file (that is, organization=sequential, recordformat=STREAM_LF, carriagecontrol=none, max = 512 bytes even if the "true" record length is bigger !!).

Of course a second problem with binary files is related to the internal representation of the content, which is machine-dependent. This means transparent transfer of the data might not be of such great use, as the data has to be converted through a program anyhow. This is in general always true for floating point data, and in general not true for integer data (when stored in standard 2-complement form); unfortunately the VAX is not using standard 2-complement form, but a non-standard form, which, however, can be reconducted to the standard one by a byte-swapping procedure (simple for 16-bit INTEGER*2).

A further consideration arises from the fact most communication modalities foresee an ASCII and a binary mode. This is called "image mode" in FTP, and -v (verbatim) mode in dnicp.

The latter modes are the only one able to preserve integrity of the transmission, and shall be used <u>mandatorily</u>, as shown also in the following table.

.cp10
$RULE1800,5

**Table b.7: format of standard binary direct access files**

$RULE1800,3
Copying a standard binary file from VAX to Sun
$RULE1800,3

| Modality | Resulting sunfile format if the mode for transfer was ASCII | binary |
|---|---|---|
| VAX FTP or FTPU | extra newline inserted at the end of each rec | data copied unchanged |
| SUN FTP | idem | idem |
| SUN NFS | n/a | idem[1] |
| SUN DNI | extra newlines inserted | data copied unchanged |
| DECNET | n/a | idem |

1: if a copy with byte swapping is desired (this means the sunfile could be used directly on the Sun, if its content is INTEGER*2) use :

```
   dd if=vaxfile of=sunfile conv=swab
```

Otherwise "copied unchanged" means the bytes remains in the same position as on the originating machine.

$RULE1800,3
Copying a standard binary file from Sun to VAX
$RULE1800,3

| Modality | Resulting vaxfile format if the mode for transfer was ASCII | binary |
|---|---|---|
| VAX FTP or FTPU | stream cc=list, record length is screwed up as imbedded 0A hex are in- erpreted as newlines | recordformat=variable cc=none data is copied unchanged however VAX has no track of the original record length (sees an unique block of undefined length |

| | | |
|---|---|---|
| SUN FTP | idem | idem |
| SUN NFS | n/a | stream, cc=none, max=32767 bytes data copied unchanged[1] however ANALYSE/RMS interprets it uncorrectly (0A hex as newlines, last record without valid deli miter) |
| SUN DNI | stream, cc=list | stream, cc=list this is <u>contrary</u> to what claimed in the DNI manual (fixed length). |
| | in both modes ANALYSE/RMS interprets it uncorrectly (0A hex as newlines, last record without valid delimiter) | |
| DECNET | n/a | if original file is shorter than 8 vax blocks, is copied as stream, cc=list, with a wrong maximum record length (as VAX FTP in ASCII mode), and is copied only partial ly. If it is longer is copied as fixed length 512 bytes, cc=none |

1: if a copy with byte swapping is desired use :

```
   dd if=sunfile of=vaxfile conv=swab
```

$RULE1800,5

## C. Consideration for file transfer of astronomical formats

Further consideration has been given to the exchange of files in disk FITS format, and in IRAF OIF format (these tests should be extended to IRAF STF format and MIDAS BDF and TBL format, as well as to the verification of consistency of the IRAF and MIDAS disk fits).
These binary files are not in the standard direct access format mentioned above, but are instead seen as :

Sun FITS plain byte streams (vax sees them as stream, cc=none, max=512)VAX FITS fixed 512 bytes, cc=none

Sun OIF plain byte streams (vax sees them as stream, cc=none, max=512) VAX OIF   stream, cc=list, max=0

It shall be further noted that IRAF FITS files are perfectly conformant to the FITS standard, that is the binary information is stored as standard 2-complement integers <u>on either machine</u>. Therefore <u>any binary copy</u> via ftp, nfs, dni or decnet is sufficient to export the files for use onto the other machine with one exception as listed in the table below :

$RULE1800,5

**Table c.1: exchange of FITS files**

$RULE1800,3

from VAX to Sun

$RULE1800,3

| Modality | Comments |
|---|---|
| VAX FTP | OK if transfer is type IMAGE |
| VAX FTPU | OK if transfer is binary |
| SUN FTP | OK if transfer is binary |
| SUN NFS | OK |
| SUN DNI | OK if transfer is -v |
| DECNET | OK but the resulting sunfile is somewhat bigger than expected (however IRAF reads it without errors) |

.cp4

$RULE1800,3

from Sun to VAX

$RULE1800,3

| Modality | Comments | |
|---|---|---|
| VAX FTP | OK if transfer is type IMAGE; | recfm=variable, cc=none[1,2] |
| VAX FTPU | OK if transfer is binary; | as above[1,2] |
| SUN FTP | OK if transfer is binary[3] | |
| SUN NFS | OK | recfm=stream, cc=none, max=32767[1] |
| SUN DNI | OK in any case | recfm=strean, cc=list [1] |
| DECNET | NO | recfm=variable, cc=none max 512 [4] |

1: format is different from native VAX FITS files, but IRAF can read OK.
2: VAX hangs up if ASCII transfer attempted for such big files
3: Sun ftp gives error if ASCII transfer is attempted (stop UCX to cure it)
4: vaxfile is slightly bigger than expected, there are spurious extra characters at beginning which make it illegible

$RULE1800,5

It  is  also  possible to access directly a FITS file  residing on  one machine's  disk from within an IRAF running on the other machine in  the following conditions :

.cp5
$RULE1800,5

**Table c.2: programmatic access to remote FITS files**

$RULE1800,3
     from VAX to Sun
$RULE1800,3

within IRAF    there  is  no  way  to access (via Decnet)  a FITS  file residing on Sun. IRAF is unable to recognise nodenames in VMS  filenames,  and  is  also unable  to interpret  VMS logical  names.  The  Sun FITS file  cannot  be accessed directly  by rfits,  nor does it work to make a cd to the sundir within IRAF (though this is possible from DCL)

$RULE1800,3
     from Sun to VAX
$RULE1800,3

within IRAF    rfits nfsfile "" irafimage             (quick and safe)where nfsfile is /vax/dir/fname

within IRAF    cd /vax/dirrfits fname "" imdir$irafimage (beware of imdir)

```
               if  imdir is not specified,  the header of the
               IRAF image will  be  created  on /vax/dir and
               the  data  file  will instead be on Sun in imdir.
               IRAF is perfectly capable of working  with images
               split among different machines  this way,  but
               this is certainly most confusing and not  what
               the user expects.  To force the header to reside
               on  Sun, specify  imdir  (or any valid sundir if
               you do  not  care having the header on a
               different directory than the data)
```

within IRAF    the same techniques allow to wfits a Sun irafimage to VAX

$RULE1800,5

It  is  instead <u>not possible</u> to exchange in an useful way the OIF  files for a variety of reasons :

they are not plain binary files (byteswapping does not work; they are a mixture at least of funny ASCII -one byte every two, separated by nulls- plain integers and double-byteswapped fields).the files are double (an header file and a "pixel" file), where the header file has encoded the pathname of the corresponding pixel file.
Each of the double file is preceded by its own internal header.

Actually the pixel data (if integer) can just be byteswapped, but this does not apply to the header. It is also not possible to use any standard mechanism of file copy, copying the two components of the double file, as the copied header (if legible on the destination machine, which is generally not the case) will still point to the original place on the origin machine !! Also doing an imcopy from within IRAF (in the case remote directories can be accessed, i.e. from Sun IRAF) will not work (as the internal representation of the origin machine will be used).

There are only two ways to effectively use a remote OIF file :

the preferred one, which gives full access, is to physically transfer the file using an intermediate FITS file

or, for alphanumeric or graphical inspection only, one can do a remote login on the other machine and run IRAF there. This has no particular advantage (and is generally more limited) than running it directly from a terminal attached to the other machine (except the fact one sits on the same chair).

.cp50

This page intentionally left blank

.pa

**D. Directory access**

There are two types of remote directory access possible : change working directory to the remote directory, and listing of the remote directory content. The second type is most widely used, and the relevant syntax is described in table d.2. The first type is possible only in the cases listed in table d.1.

The syntaxes given here are in the default case (no proxy accounts, logical names, etc.) and are subject to the same simplifications described in table b.1 above.

$RULE1800,5

**Table d.1: syntax for changing working directory**

$RULE1800,3

from VAX to Sun

$RULE1800,3

```
DECNET only:   SET DEF sun"user pwd"::fsdir:[dir.dir]
               also it is possible to use a logical name
```

$RULE1800,3

from Sun to VAX

$RULE1800,3

```
NFS only:      cd /vax/dir/dir
```

$RULE1800,5

The syntax to do a directory listing is presented in the table below (of course directory listing of a remote directory must by definition be initiated on the local machine !).

$RULE1800,5

**Table d.2: syntax for directory listing**

$RULE1800,3

from VAX to Sun

$RULE1800,3$ELISM

| Modality | on host | command(s) to be given | Rating |
|---|---|---|---|
| VAX FTP | vax | FTP \|connect sun\|user\|pwd\|cmd[1]$ELISM "sunfile"\|...\|exit | messy |
| VAX FTPU | vax | ftpu\|open sun \|user\|pwd\|cmd[2]$ELISM sunfile \|...\|quit | good |
| DECNET.1 | vax | directory sun"user pwd"::psefile[3]$ELISM | fair |
| DECNET.2 | vax | directory sun"user pwd"::"sunfile"[4]$ELISM | messy |

$ELITE

1: cmd can be: dir or dir/full for short and long listing. The short listing is a multicolumn listing. The long listing is still limited to one line

2: cmd can be: ls or dir for short and long listing (as above) It is not possible to specify a file to receive the output (as foreseen in the Unix standard ftp)

3: note that / shall be defined as a fsdir root:Usage of VMS qualifiers possible, results not guaranteed correct.Does not list Unix hidden files (name starting with .)One cannot (with [-] and [--]) go above the login directory of the user specified. One cannot specify the root user and password as access control string. However one can access / also from below as e.g. usr:[-]

Does not recognise correctly filenames, and a single directory may appear incorrectly split in subdirectories at "odd" (from VMS point of view) filenames.

4: Has not the above problem, but the full pathname is always printed, and within quotes (painful). A wildcard of * shall be specified to list the content of a directory.

$ELITE$RULE1800,3
from Sun to Vax
$RULE1800,3$ELISM

| Modality | on host | command(s) to be given | Rating |
|---|---|---|---|
| SUN FTP | Sun | ftp\|open vax\|user\|pwd\|cmd[1]$ELISM vmsfile\|...\|quit | good |
| SUN NFS | Sun | ls[2]$ELISM nfsfile | ideal |
| SUN DNI | Sun | dnils –u user –p pwd 'vax::vmsfile'[3]$ELISM | messy |

1: cmd can be: ls or dir for short and long listing. The short listing is a multicolumn listing. The long listing is a VMS DIR/FULL, which may be prohibitively long !!!
2: ls can be replaced by any valid alias (or have any valid option)There is some (reasonable) remapping of VMS file names in the case of files with more than one version (see UCX documentation).
3: there are funny switches to include (without guarantee) extra information in long listings
$ELITE$RULE1800,5

## E. Other access of remote file at session level

There are other ways by which it is possible to access a remote file at the level of the command interpreter (DCL or csh; programmatic access is described in section F). These include typing, printing, editing, renaming and deleting.

They are generally subject to the same limitation of access described for file transfer. Of course most of these activities are interactive, and therefore require an interactive login (not an ftp login).

.cp10
$RULE1800,5

**Table e.1: typing remote files**

$RULE1800,3

| | | |
|---|---|---|
| DECNET.1 | on vax | type sun"user pwd"::psefile |
| DECNET.2 | on vax | type sun"user pwd"::"sunfile" |
| SUN NFS | on sun | cmd[1] /vax/nfsfile |

1: use any valid typing command (cat, more) or alias
   the command works only if the vaxfile is in STREAM format

$RULE1800,5

$RULE1800,5

**Table e.2: printing remote files on local[2] printer**

$RULE1800,3

| | | |
|---|---|---|
| DECNET | on vax | access to remote files not supported copy to local temporary file and print locally |
| SUN NFS | on sun | lpr[1] /vax/nfsfile |

1: or any valid aliasthe command works only if the vaxfile is in STREAM format
2: printing local files on remote printers is subject to local arrangements.

$RULE1800,5

$RULE1800,5

**Table e.3: editing remote files**

$RULE1800,3

| | | | |
|---|---|---|---|
| DECNET.1 | on vax | edit sun"user pwd"::psefile | (1) |
| DECNET.2 | on vax | format not supported by edit | |
| SUN NFS | on sun | textedit /vax/nfsfile | (2) |
| | | vi /vax/nfsfile | (2) |

1: works OK, however does not create backup version, and always overwrites the new file (no new version created)
2: only STREAM files supported, textedit opens the file correctly, and so does vi (which however puts the file in readonly mode). However it is not possible to save the changes

(as the editor is "unable to backup") : the only way is to do a "store to new file".

$RULE1800,5

.cp10
$RULE1800,5

**Table e.4: deleting remote files**
$RULE1800,3

| | | |
|---|---|---|
| VAX FTP | on vax | FTP \|connect sun\|user\|pwd\|delete "sunfile"\|..\|exit |
| VAX FTPU | on vax | ftpu\|open sun \|user\|pwd\|delete sunfile \|..\|quit |
| DECNET | on vax | network operation not supported |
| SUN FTP | on sun | ftp \|open vax \|user\|pwd\|delete vaxfile[1] \|..\|quit |
| SUN NFS | on sun | rm[2] nfsfile |
| SUN DNI | on sun | no command available to do it |

1: you shall specify the version number
2: or any valid alias

$RULE1800,5

$RULE1800,5

**Table e.5: renaming remote files**
$RULE1800,3$ELISM

| | | |
|---|---|---|
| VAX FTP | on vax | FTP \|connect sun\|user\|pwd\|rename "sunfile" "sunfile"\|..\|exit |
| VAX FTPU | on vax | ftpu\|open sun \|user\|pwd\|rename sunfile sunfile \|..\|quit |
| DECNET | on vax | network operation not supported |
| SUN FTP | on sun | ftp \|open vax \|user\|pwd\|rename vaxfile vaxfile[1]$ELISM \|..\|quit |
| SUN NFS | on sun | mv[2]$ELISM nfsfile nfsfile |
| SUN DNI | on sun | no command available to do it |

1: operation is performed, but ftp hangs up (UCX shall be restarted)
2: or any valid alias. BEWARE mv is not a plain rename, but also moves directory to directory. If a full pathname on vax is not specified for the new name, you will find the file on your current sun directory !!

$ELITE$RULE1800,5

.cp50

## F. Fortran access to text files

This section is concerned with the access to files (standard text files as defined above) from within a Fortran program, and using standard Fortran statements. The files to be accessed may be remote files, or local files which have been transferred from a remote machine by any of the communication mechanisms.

$RULE1800,5

**Table f.1: Fortran access to text files**

$RULE1800,3

| | | | |
|---|---|---|---|
| from vax | reading | a remote file sun"user pwd"::psefile | is OK |
| | reading | a file transferred by any way | is OK |
| | writing | a NEW remote file | is OK[1] |
| | appending | to an existing remote file | fails[2] |
| | appending | to a file transferred by any way | is OK |

inquiring the file organization of a file returns :

| | |
|---|---|
| for a remote file | unknown, cc=unknown |
| for an ftp transfer | stream, cc=list, max = 0 |
| for an nfs transfer | stream, cc=none, max = 32767 but it is modified on open !!! becomes stream, cc=list, max = 32767 |
| for a dni transfer | stream, cc=list, max = 0 |
| for a DECNET transfer | stream, cc=list, max = nn |

1: the new sunfile belongs to group wheel.
2: "network file transfer mode precludes operation"

$RULE1800,3

| | | | |
|---|---|---|---|
| from sun | reading | a remote file /vax/nfsfile | may be OK[1] |
| | reading | a file transferred by NFS | may be OK[1] |
| | reading | a file transferred by any other way | is OK |
| | writing | a NEW remote file | is OK |
| | appending | to an existing remote file | may be OK [2] |
| | appending | to a file transferred by any way | is OK |

inquiring file organization is not applicable to Sun

1: is OK for stream files, fails for standard VAX text files

2: is OK for a stream file with cc=list or cc=none, any other organization gives crazy, not easily reproducible results, including the apparent disappearance of files (requires dismounting the nfs file system and remounting it again). NO new version is created anyhow.

$RULE1800,5

.cp3
Please note that on VAX it is required to specify CARRIAGECONTROL='LIST' explicitly in the OPEN statement for a NEW file (the default is Fortran carriage control, which is not nice). If a stream file is wished, add also RECORDTYPE='STREAM-LF'. An existing file will always be opened automatically in a correct way.

## G. Fortran access to binary files

This section is concerned with the access to files (standard binary direct access files as defined above) from within a Fortran program, and using standard Fortran statements, similarly to what done in section F for text files.

However the excessive variety of file organizations on VMS, and the oversimplified file organization of Unix, make the exchange of such files problematic.

The main problem areas are the following:

for vax: the usage of the RECL keyword in VAX Fortran is inconsistent and out of standard. INQUIRE returns the correct RECL in bytes if the file is unopened, however an OPEN statement for unformatted output wants the RECL in 4-byte units !!

the use of direct access i/o is possible only on files which have recordformat=fixed. (This as such is not a severe limitation, and is customary under many systems; however it may give problems when the file accessed or transferred over the network has not such organization ... note that the organization referred here is not the physical arrangement of records, which may be OK, but the information written in the VMS directory area !!).

for sun: the record length information is not associated in any way to a file directory entry (INQUIRE returns zero in all cases)

actually any file can be read specifying any RECL and
is up to the user to specify a consistent one

$RULE1800,5

**Table g.1: Fortran access to binary d.a. files**

$RULE1800,3

| | | | |
|---|---|---|---|
| from vax | accessing | a remote file via DECNET | does not work |
| | accessing | a transferred file | does not work[1] |
| | inquiring | it's not possible to determine the correct recl | |

1: the only exception is a file transferred with DECNET, when the RECL is exactly 512 bytes (the file can be read in any case forcing such RECL value if it is bigger than 8 blocks)
See below for notes on file format conversion.

$RULE1800,3

| | | | |
|---|---|---|---|
| from sun | reading | a remote file via NFS | is OK |
| | writing | a new remote file via NFS | is OK[1] |
| | writing | to an existing remote nfsfile | does not work |
| | accessing | any transferred file[3] | is OK[2] |

1: however this file is illegible on VAX by a similar Fortran program
2: RECL shall be specified by the user to read an existing file
3: the file shall have been transferred in a binary mode

$RULE1800,5

I have been unable to find a simple way (with a single command, e.g. COPY or CONVERT/FDL) to change the file organization of a vaxfile generated via file transfer from sun from whatever it is to fixed length (even if the data are OK).

This could possibly be done by a dedicated program.

A way exists to force a correct transfer of direct access files from Sun to VAX (that is to build the vaxfile as recordformat=fixed), and is to make use of the FDL facility provided by UCX (**only** in the FTP VMS-style, which is not the handiest). This facility has actually been designed for another purpose, that is exporting a file from VAX to Sun and recovering it back.

To do this for each vax file two files are copied to Sun, the data files (forced to binary mode), and an associated FDL file.

Note that there is a one-to-one correspondence in the file names (and a consequent proliferation of FDL files on the two machines). When the file is reclaimed back to VAX (this has to be done by a VAX FTP) both files are brought back and the FDL file is used to derive the info on recordformat and recl.

This can be <u>adapted</u> to work for transfer of a native Sun direct access file setting up utilities which:

> on Sun build an FDL file (or update an existing template) with the fields relevant to the file to be transferred (namely the recl)the FDL file shall also be assigned the appropriate name !!
>
> on VAX issue a FTP command with FDL option to retrieve the sunfile

As it can be seen the procedure is awkward as it requires coordinated actions on the two machines; it is also not general enough, as it depends on a feature of UCX which is not standard in ftp.