.pl60
.mb 3
.po 0
.he$ELISM The UNIQ command interface for VMS and Unix users - Definition document$ELITE
.fo$ELISM The UNIQ interface - Definition Document - Dec 1990 - Page #$ELITE$FF

The UNIQ interface

An unique command interface for VMS and Unix users

Definition document

prepared by L.Chiappetti - IFCTR
Dec 1990

0. Purpose

> The purpose of the **UNIQ** interface is to define and implement an uniform way by which users of different operating systems may access the more usual commands exactly in the same form, so that they can get accustomed to only one syntax.

This will prevent annoying confusions when one has to remember many different command forms. Morevover it will allow a choice of the command syntax which is more natural and comfortable than the one used by some operating system (although this is somewhat subjective and linked to the preference of the authors).

There are two documents related to the **UNIQ** interface. This is the definition document (an earlier draft dated Aug 1990 was discussed with B.Garilli, whose comments are gratefully acknowledged), which collects the requirements and guidelines, and is not intended for general release. The second document is the users guide, which should be available to the general user.

The guidelines used are the following:

> Only the more used, standard "file manager" commands will be affected.
>
> There will be almost no attempt to alter the file naming (this is not a programmatic interface of the type provided e.g. by the IRAF VOS or the proposed XAS VOS).

The new commands shall be implemented as far as possible with plain synonyms (Unix csh aliases or VMS symbols) or by simple procedures (Unix shell scripts or VMS .COM files) at command interpreter level (e.g. DCL or csh). They will not be implemented via higher level, compiled language (e.g. Fortran) programs.

The starting implementation will consider VMS and Unix under csh as target systems. The syntax will be chosen as an intersection (in the authors' taste) of VMS and Unix, to be as close as possible to natural language. MS-DOS will also be considered as an example. The above statement means that in general a command should be a pronounciable verb, even if long (DELETE is preferred over rm); and that the presence of "funny" characters (quotes, square brackets, slashes etc.) shall be avoided (unalias is preferred over DELETE/SYMBOL).

.cp5

1. Logging in

1.1 Password and other security features

The setting up of an account is subject to a number of local (site-dependent) arrangements. Therefore they are not described here (for IFCTR refer to the announcement in "Nuove facilities di calcolo presso IFCTR", and to the Blue Book).

The only control the general user will have about its own account, is the choice of the password. This is done by commands **SET PASSWORD** (VMS) and **passwd** (Unix). These commands are used infrequently enough, that is not worthwhile to define an UNIQ command.

1.2 Welcome and other system messages

It is desirable to have most of the site dependent arrangements which should apply to each session (inclusive of the definition of the UNIQ interface itself !!) to be handled by a systemwide login procedure.

The exact content of such procedures is site-dependent and is described elsewhere (see refs. in 1.1 above). These things are however arranged differently in VMS and Unix systems (mostly because Unix systems are generally "personal" or "single user" workstations). A common arrangement is reached as

follows :

Definitions which are permanent (e.g. system logical names referring to disks, or permanent server processes) are better initiated at boot time. This is done in VMS by the SYSTARTUP.COM and SYSTARTUP_V5.COM files, and in Unix by the rc, rc.boot and rc.local scripts.

Session definitions, applying to all users, are handled in VMS by the SYLOGIN.COM file. There is no equivalent of this in Unix: a reasonable solution is to have each user's .login and .cshrc files linked to a common version. This way a "system login" is actually executed first. See 1.3 below.

1.3 User defined login commands

Normally an user expects to be able to have a personal, customized login procedure (a set of commands executed each time the user logs in). In Unix one has in addition the distinction between some commands which have to be executed in each shell, and some others only in the login shell. The normal situation sees the files :

VAX:	LOGIN.COM
Unix:	.cshrc (each shell) and .login (login shell only)
DOS:	AUTOEXEC.BAT
IBM:	PROFILE EXEC
HP:	user HELLO file

.cp2
Due to the arrangement of 1.2 above to simulate a systemwide login procedure in Unix, the user cannot do personal changes to .login and .cshrc. A file .mycshrc (called immediately by .cshrc before exiting) is provided to perform the function of user customizable login file.

1.4 the prompt

One nice feature offered by most operating system, is the possibility of customizing the system prompt. In systems with an hierarchy of subdirectories, it is quite handy to have the current directory displayed as part of the prompt. This is done very easily in MS-DOS.

The design goal is to provide the same facility as in MS-DOS,

i.e. to have (part of) the path displayed and updated automatically when the directory is changed. However, while the first part is easy in all systems, the second (automatic update) is NOT provided by VMS or Unix.

To provide it one is forced to alter the "change directory" (see 3.5) command to redefine the prompt.

The current choice is to have the prompt defined as follows :

> VMS: the last two components of the current path
> Unix: the last component of the current path and the command number

The implementation given is different. Therefore it is also different the corrective action an user should take to define a different prompt (e.g. the full path name, or a fixed prompt), in the case one does not like the default one.

> In VMS a global symbol PROMPT is called by the CD command (see 3.5). The UNIQ choice defines this PROMPT to point to a particular .COM file. The corrective action in the user LOGIN.COM is to redefine this symbol to point to an user .COM file, or to alias a fixed SET PROMPT command.
>
> In Unix the prompt definition is part of the redefinition of the cd command (see 3.5) as an alias. The corrective action in the user .mycshrc is to re-alias cd to a string similar to the one in .cshrc, but with a different prompt setting.

1.5 Getting help

Access to the system help files shall occur via an UNIQ command. The modalities of the presentation of the help file will be system dependent (use system's native facilities). The proposed syntax is:

> **HELP** topic

This command is native in VMS (also in IBM VM/SP);
an alias with **man** is necessary in Unix.

2. File names

2.1 Elements of file names

The syntax of a complete file name is quite different on the various systems :

```
VMS:      node::disk:[directory]fname.ftype;v
Unix:     /filesystem/directory/fname
DOS:      disk:\directory\fname.ftype
IBM:      fname ftype fmode
HP:       fname:sc:cr
```

there will be no attemp at command interpreter level to provide an unique filename mapping with a common syntax. System dependent file names will be used. Also the wildcard mechanism will be used in system dependent way.

As recommendations one might say that :

> Unix users are encouraged to use for their files (for system-dependent files one has to live with whatever convention is in use) a single dot in the filename, and to use this consistently with an interpretation of the form fname.ftype.
>
> VMS users are encouraged to keep a single version of their files. If for instance one defines a version limit of 2 for the user directories, this induces a behaviour similar to what is common in Unix (that is two versions of a file, like Unix name and name%, the current and the backup one). The second version could then be purged as soon as possible (at logout or at next login).

2.2 Types of files

The concept of file type has different meanings. On one hand most systems (Unix being noticeable as not doing that systematically) use a ftype item in the name to identify families of files or similar usage. This is covered by the last recommendations in 2.1.

Not necessarily related with the naming, files may have different structures or organizations (variable and fixed, sequential and direct, etc.). This is a very fishy area, of concern to programmers : some elements in this sense are available in separate documentation ("A report on tests of

VMS-Unix file exchange", Aug 1990, by L.Chiappetti).

2.3 Security of files

Protection of files is arranged in different ways (based on disk protections on systems like IBM or HP, the latter with the additional security code; based on an access mask on systems like VMS and Unix, unfortunately with different modalities).

One set of recommendations may include :

System disk area will be left alone (system-dependent protection).However it will be preferred (e.g. for all public system utilities, as well as for the astronomical packages) to have publicly readable areas (read-and-execute access for all).

Truly public data areas will have read-and-write access for all

Private areas, according to sense of privacy of the owner, may have either read-and-execute for all, and write for the owner; or no access for others than the owner.

As far as possible the following mapping of protections shall be enforced between VMS and Unix systems :

.cp4

VMS OWNER and SYSTEM shall be the same as Unix user.
VMS GROUP shall be the same as Unix group
VMS WORLD shall be the same as Unix other
GROUP and WORLD (group and other) shall be the same (groups shall be ignored for our purposes, as they are an annoyance)In general write and delete access shall go together.

At the level of the UNIQ interface there could be commands to :

set the default protection	(SET PROT/DEFAULT	or umask)
change file protection	(SET FILE/PROT	or chmod)
change file owner	(SET FILE/OWNER	or chown)

However the first command (setting file protection) is seldom necessary, as this will be arranged in the systemwide or user login. The last command (change owner) is restricted to SYSTEM (or root), and not available to the general user. In both cases there is no need for an UNIQ command.

One is therefore left with a single set of commands to change file protection, preferably using a rwx nomenclature and not a a 777 nomenclature. The first two are recommended, and the last should be unnecessary (the rule is group=world for what protections are concerned).

protuser prot set Unix u, VMS S+O to prot
protother prot set Unix go, VMS G+W to prot
protgroup prot set Unix g, VMS G

.cp20

3. Disk space organization

3.1 Directories and disk sections

One first difference is between systems oriented towards fixed disk sections (primarily IBM virtual disks or HP cartridges), and systems oriented towards freely-growing directories (DOS, VMS and Unix). The latter three share of course a fixed-size limitation given by the physical disks (or partition in Unix) size. However in VMS (and DOS) the disk volume is an independent part of the file name, while in Unix the filesystem (either disk or disk partition) can be mounted anywhere in a tree.

It would be desirable, at least for some purposes, to adopt an unique logical scheme similar to Unix, but possibly this goes beyond the UNIQ interface.

Also it is not proposed to change the directory naming in general, but only for applications like the "cd" command (see 3.5 below). The extent of this being applied to other commands is worth discussing. In general a mapping of the type described in 3.5 requires a dedicated command procedure (.COM file or shell script), possibly through a common service procedure, and forbids use of plain aliases (or VMS global symbols).

3.2 Directory listing

There are many variations on the format of a directory listing. The proposal for the UNIQ interface is to select one of those as default and access it via the following command :

DIR pathname

This command can be implemented as an alias (symbol) in Unix and VMS, provided one accepts to use system dependent pathnames.

All other forms will be accessible using the system dependent commands (DIRE*CTORY in VMS and ls in Unix). So far the pathname is in system dependent format (at a later time one might think to apply here too the scheme 3.5 below for naming).

It could also be nice to have a DIR which produces an unique format which is system independent (typically one which indicates filesizes in kbytes and not system-dependent blocks, or which treats filenames and types as separately aligned fields), but this requires at least a procedure and possibly a program.

It would also be desireable to have directories sorted by the parameters, particularly by file types.

Another desirable feature, which however requires a program, is an utility similar to the IBM FILELIST or DOS PCTOOLS (full screen display with possibility of selecting files for operation).

.cp5

3.3 Disk space available

As a second priority proposal for the UNIQ interface are commands to:

get the total disk space occupied and free (also as percentage of the total, similar to IBM Q DISK) for a physical disk (proposed command name **qdisk**)

same for a directory (% of quota if setup ?); proposed command name is **qdir**.

a command (cfr. EXEC SIZE on IBM) which returns just the number of kbtyes for a family of files. Proposed command name **size**.

All the above commands should return the quantities in kbytes.

3.4 Access to new disk sections

While access to new disks on fixed-disk systems is complex and generally possible for temp space only (cfr. IBM DEFINE,

ACCESS and LINK; or HP AC, MC; cfr. also DOS SUBST), this is a common operation for hierarchical directory systems, and is candidate for an UNIQ interface command with syntax:

mkdir directory

This is native in Unix, and can be aliased to CREATE/DIRECTORY in VMS. I propose to set up also some directory characteristics at this time if possible (e.g. /VERSION in VMS). TBV if normal file protections are OK for directories.

So far the directory name shall be specified in system dependent format (only later in the 3.5 format).

3.5 Moving between disk sections

One has first of all to know on which disk a file will be sought by default, and possibily a way to change this. For fixed-disk systems this is not always possible, or is quite rigid (not on HP, yes on IBM changing a disk mode), while for directory-oriented systems is a common operation. The proposal for the UNIQ interface is to have two commands:

cd directory	to change the current directory
pwd	to know which is the current directory

Although two such command are available in Unix, they need rewriting. The reason for that is to make them emulate the DOS CD command, which, in addition to the change of directory will also reset the current prompt (if arranged to display the working directory). This does not occur in Unix. However it can be arranged easily via aliases.It could be possible to alias the two commands in VMS with SET DEF and SHO DEF. The latter is OK, but the former is not acceptable. One reason is the fact that SET DEF does not check on the existence of the target directory, and VMS complains later. The other is that the directory syntax of VMS is annoying. A .COM file is therefore mandatory.

The ideal directory naming would be the Unix one, however this cannot be emulated at DCL level because slashes are special to it (qualifier delimiters). A different separator (dot or backslash) should be used.

The following Unix features should be propagated into VMS (appropriate separator substituted to Unix slash) :

```
/    to map root (DUA0:[000000] ? what about other disks ?)
/dir absolute path, better than VMS [dir]
dir  relative path, better than VMS [.dir]
.    current directory, better than VMS [] (unnecessary ?)
..   parent directory,  better than VMS [-]
```

Additionally a "blank" CD command will return to the home directory.

One should also mantain the possibility of using a VMS logical as pointer to a directory, and one is forced to keep the separation between default device and directory. An improvement w.r.t VMS SET DEF is to have a preliminary check of the existence of the target directory, which is missing in VMS (with great annoyance).

The following VMS features are absent in Unix, and will be renounced :

```
grandparent          [--]
all below current    [...]
all on volume        [*...]
all one step below   [*]
etc.
```

A further question concerns the setting up of a default path for executable files (customary in Unix and DOS, but unavailable in VMS), but this is beyond the UNIQ interface.

3.6 Losing access to disk sections

On fixed disk systems this might mean to hide a disk temporarily (IBM RELEASE) or permanently (IBM DETACH, HP DC). On a directory-oriented system it means deleting the directory and the files below it.

The proposal for the UNIQ interface is a command :

rmdir directory

If it is desired that this command automatically deletes all files in the directory (i.e. "unconditional removal"), in Unix it is necessary to set up an alias to rm -r. In VMS probably one should use a small .COM file (first change protections, then DELETE ... as directories are normally delete-protected in VMS; look for examples in VMS manuals).

.cp10

4. Handling files

4.1 Creating a file

It is not certain whether an utility to create a file without use of an editor or program is needed. Were it desired, it is possible to emulate the VMS CREATE command in Unix doing a cat >file. This will not be implemented in the UNIQ interface.

.cp5

4.2 Copying files

The basic proposal for the UNIQ interface is to have one simple command with an uniform syntax like :

COPY file1 file2

This is native in VMS, and a simple alias of cp in Unix. However all peculiarities of the different systems are still present.

At a later time it would be desirable to handle uniformly within the UNIQ interface also the "append" and "replace" functions of the copy command (in IBM terminology). The "replace" function could easily be aliased to an **OVERWRITE** command.

Finally it could be desirable (but possibly can be done only with a program) to be able to specify a starting and ending record for the copy (again the example is the IBM COPYFILE command).

4.3 Deleting files

In this case the proposal for the UNIQ interface has the syntax:

DELETE file

This is done in Unix with an alias of rm, and in VMS with a small procedure (the main purpose of this is to force a wildcard-version ;* when no version is specified).

It is suggested to use rm -i as alias in Unix : this will ask permission for deletion. This is useful due to the "typeless" (and confusing) characteristic of Unix files, to

prevent unwanted deletions.
In VMS the usage of DEL*ETE as alias will prevent use of the normal VMS DELETE command, unless this is first saved to some other alias. It is to be verified how to preserve usage of the qualifiers (e.g. /CONFIRM) and pass them to the .COM file.

.cp5

4.4 Renaming files

The obvious proposal for an UNIQ interface for this command is :

RENAME oldname newname

However in both Unix and VMS the corresponding commands (candidate for an alias) are mv and RENAME, which performs not only a plain renaming, but also move the directory entry of the file (still pointing to the same physical file), which effectively moves the file to another directory.

If this feature is NOT wished as normal behaviour for RENAME (if one does not specify a path for newname, the risk is to move it to the current directory !!), a special procedure shall be replaced !!

4.5 Moving files

If the "move" feature is disabled for the UNIQ RENAME command, a separate command **MOVE** could be implemented as an alias of the "normal" (sic!) behaviour of mv and RENAME (as suggested in 4.4).

(on fixed-disk systems this is a copy followed by a delete)

.cp5

4.6 Others

A needed UNIQ interface command is required for typing a file :

TYPE file

This shall be aliased in a way to stop display at the end of the current page (more behaviour under Unix, TYPE/PAGE under VMS), and possibly to avoid screwups of the terminal if non-printing characters are in the file (a cat switch piped into more in Unix, VMS ???).

A Browse-like utility could also be useful (typically implemented by an edit with a READONLY flag).

A way of looking at a file in hex is also useful (an UNIQ alias could be made of commands like Unix od or VMS DUMP, but there are many variants here; possibly only a reduced subset, e.g. HEXTYPE, should be implemented in the UNIQ interface).

Possibly some form of UNIQ interface (without affecting the way the system-dependent programs work) could be sought for :

file comparing :	diff and DIFF	(unnecessary)
pattern search :	grep and SEARCH	
sorting :	sort and SORT	(unnecessary)
protections:	see above 2.3	
other file chars:	in VMS SET FILE	(does not apply to Unix)
secondary names:	Unix ln (can be done in VMS SET FILE ?)	

There is a bunch of functions in VMS (UNLOCK, ANA/RMS, SET RMS, CONVERT, CREATE/FDL, EDIT/SUM, EDIT/FDL) which are system dependent, but should be looked in detail.

5. Terminal characteristics

5.1 Keyboard characteristics

I propose that system-dependent ways of defining user keys etc. are left alone and not made part of the UNIQ interface.

.cp3
For the rest it looks like that mapping of keyboards to force a decent behaviour of VAX terminals (insert toggle, delete-character in correct order, that is right of the cursor position, etc.) is not possible.

The use of arrow keys to recall the commands is acceptable. A similar facility for Unix shall be appreciated (better than a !!:p and editing with the mouse cut-and-paste), but possibly requires too deep a change to the shell.

It is possible to alias the VMS RECALL command with **HIS** or **history** (since it performs similarly to such Unix command). Conversely one could have an Unix alias **recall** to recall and edit the n-th command.

It is a bit annoying that EOF is ctl-D in Unix and ctl-Z

in VMS, particularly as ctl-Z puts an Unix process in "stopped" status.

.cp4

5.2 Screen scroll etc.

This feature is terminal dependent, and is available in a decent way only on HP terminals and Sunview windows.

5.3 Querying and setting terminal characteristics

It could be possible to define an UNIQ interface to the commands used for terminal setting and status querying (SET TERM and SHO TERM in VMS, tty, stty, tset in Unix). However these commands are of rare use and complicate syntax. Possibly a plain **QTERM** alias for the "query terminal" functions is enough.

5.4 Console spooling

The possibility of recording a terminal session in a file is offered in a handy way only by IBM VM/SP. However in a window environment like Sunview one has full scrolling functionality as well as the possibility of doing cut-and-paste of a window content into a file. In VMS the only way to do a console spool is a self-SET-HOST with LOG.

No action for UNIQ interface.

6. Querying system characteristics

The following commands which ask system characteristics could be unified under the UNIQ interface :

query the time **time** (SHOW TIME in VMS, date in Unix)
set the time (not necessary, system manager only)
query CPU time (not necessary, cntl-T in VMS, not available in Unix)

running programs **status** (aliased to some variant of ps in UNIX, possibly to SHO SYS in VMS, see also SHOW PROC/SYS STATUS, MONITOR

offing program by name (this may be some combination of status and grep in Unix, where one need the process id

ti kill a process, this has no sense in VMS due to the different definition of a process)

.cp7

7. Programming

All this topics (for Fortran programmers) are handled in a separate document as the "Real Programmer Tool" interface.

8. Editing

The only action at UNIQ interface level should be to call EDIT the default editor (the most reasonable one). This choice is left to individual user arrangements, except for the definition of a set of suitable aliases for Sunview's textedit.

Some key and command mapping shall be possible at least with some editor (via initialization files).

9. Printing

In line of principle PRINT is an obvious candidate for the UNIQ interface, however if one wants to control things like printing in different fonts and orientations etc. a different arrangement may be desirable.

The proposed solution sees on Vax a number of commands for the different fonts : **ELITE**, **SMALL** and **LANDSCAPE**. For the time being, while Sun has no dedicated printer, the **vprint** command is used to route prints to the VAX LN03.

Possible future improvements may concern a set of special aliases to handle "file listing" prints (that is, prints in which the first page - or all pages - are identified by an header with the file name).

Other candidates for an UNIQ interface are commands to :

look at print queue **QUEUE** (VMS SHO QUE, Unix lpq)
remove print job **DEPRINT** (VMS DEL/QUE..., UNix lprm).

Similar functions might be required for batch queues.

I would leave out of the UNIQ interface the management of print queues. This is so far important only for VMS, and involves a number of commands: ASSIGN/MERGE (DE)ASSIGN/QUE DEFINE/ and DELETE/ CHAR and FORM, DELETE/QUE, INIT/QUE, SET DEV (SHO), SET PRINTER (SHO), SET/QUE SHO/CHAR SHO/FORM START/QUE/MAN (and STOP)

10. Sending signals to system

It is difficult to find an unique way to interact with a running program, or with the system while the program is running (in the sense one does on HP BREAKing a program, or replying to SS=nn prompt).

There is no space for the UNIQ interface anyhow.

Other areas regarding interaction with the system are:

a) the schedule of job at deferred times, or background jobs

HP: programmatic via EXEC calls;
IBM only disconnected mode;
UNIX: at and & commands, crontab;
VAX: RUN process, SET PROCESS, CANCEL, STOP, SYNCH, WAIT or programmatic)

b) the use of batch jobs

partially overlaps with the above (Unix at; VMS SUBMIT; see also JOB, PASSWORD, SET OUTPUT, SET RESTART); involves all queue handling see 9.)

c) running in disconnected mode

Native in IBM; programmatic on HP; VMS SPAWN+ATTACH+CONNECT+DISCONNECT; Unix nohup.

All the above features are used seldom, mainly for system administration, therefore they will not be considered by the UNIQ interface.

.cp5
11. Communicating with users

First of all there are two obvious candidates for an UNIQ interface for a function available on most systems :

getting logged user names:	**WHO**	VMS SHO USER; Unix who
identifying oneself:	**IDENTIFY**	Unix who am i; VMS TBD

11.1 One-line messages

One-line messages are a useful, non-obtrusive way of sending a brief message to an user.

It looks like there is no obvious way of sending one-line messages (like the IBM TELL) on VMS (only operator can do it) or Unix (write allows sending multi-line messages). Nevertheless a TELL function would be useful.

11.2 Chatting

The "normal" way of talking with other people is a CHAT-like interface, as provided by VMS PHONE or Unix talk (at least the names could be unified at the level of UNIQ interface), which is however a bit cumbersome.

.cp4
11.3 Mail messages

This is handled in a separate document ("The Link Guide. An Overview of networking within IFCTR computers") and is not covered by the UNIQ interface.

11.4 Exchanging files

This is also handled in separate documents (see mainly the "Link Guide" quoted above, and references therein).

There are some different philosophies for what concerns file transfer :

apparently simpler is the remote file copy (as provided by NFS or DECNET). The copy can be initiated at either end

("get" or "put"), but this is transparent only if there are no security blocks, otherwise it becomes annoying to supply username and password, and it might not be possible to send a file to somebody without specific arrangements.

the IBM scheme (which allows only "put" or sendfile operations) uses an intermediate spool area. This is very comfortable since one can send a file asynchronously, without problems of password, and without cluttering the user permanent area. An emulation of this could be nice.

the ftp scheme requires a sort of login on the remote node, and therefore shares the problems of the first philosophy. This is however the best solution for generic file exchange.

It is proposed to leave this matter out of the UNIQ interface. It could be possible to provide some simple aliases on a case by case basis (tovax, tosun, etc. ...). This is better handled by each individual user according to one's specific needs.

11.5 remote login

It is not worthwhile to arrange an UNIQ interface for the different remote login modes (SET HOST on VAX, rlogin, telnet on Unix), but more a simple set of aliases (or even dedicated captive accounts) for the most common nodes to be reached.

This is covered in detail in the document "The Link Guide" quoted above.

11.6 inquiring on network

I do not know any easy way in TCP/IP to know whether a node is there (except the Unix ping command), what is the path to reach it and so on. The equivalent of ping under VMS UCX is accessible to SYSTEM only.

The VAX SHO NET should offer some capability in this respect, however I believe finding the exact path to a node requires lot of interactions with NCP (similarly to the RSCS SMSG in Bitnet). While it could be useful to provide user access to some NCP functions, it is not worthwhile to have a dedicated UNIQ interface.

.cp6

11.7 message control

There are functions (like IBM SET MSG, VAX SET BROADCAST, Unix biff and mesg) to control whether an user wants to allow reception of incoming messages, "phone" calls and mail notification. However it is not worthwhile to define an UNIQ interface for these.

12. Handling tapes

12.1 tape control

It would be desirable to have an UNIQ interface command of the form :

TAPE command drive

where the drive name may be system dependent, and the commands shall be of the form REWIND, FSF, FSR, BSF, BSR, EOF (to rewind, skip file/records and write tape marks. (cfr. IBM TAPE or HP *TAPE). VMS: SET MAGTAPE, Unix: mt. The drive name could in line of principle have a fixed default for the case a single tape drive is available. (A command would be necessary to set the tape density, in Unix this will interact with with the choice of the drive name). The default drive name should be kept in some environment variable.

The UNIQ interface could also handle tape allocation and mounting (ALLOC, MOUNT/FOR) and dismount (with or without unload : DISMOUNT, DEALLOC) on VMS.

12.2 system backups

By definition each system has its own way of doing tape backups (HP: SAVER/READR; IBM TAPE DUMP/LOAD/SCAN; Unix tar,cpio; VAX BACKUP) and there is no sense in providing an unique interface for what is intrinsically different.

12.3 file copying

It would be interesting instead to test whether there is a way to copy files from/to disk to/from tape in a plain way (like IBM's MOVEFILE; or Unix dd), handling also blocking/reblocking.

And also (provided there are two tapes) a simple way of copying

tape-to-tape (all, or by files) will be useful.

13. Logging out

Obviously the UNIQ interface should define a single LOG command for this. What actions are performed is system dependent (Unix foresees a .logout file to be executed then, but other systems like VMS may require a procedure to be written and aliased to the LOG command).

.cp6

Appendix

The number of commands in different systems according to their manual or help files (normal users use only the first column; the second column is for system management; the third column are programming commands used in procedures, shell scripts, .COM files, EXECs etc.; the fourth column cannot be classified of common use.

machine	all users	system only	procedures	other	total
HP	43	26	7	–	76
IBM	88	34	16+exec	58	196
Unix Sun	125	175	50	300	>650
VAX	70	65	28	4	170
UNIQ	31	n/a	n/a	n/a	31

The UNIQ interface

An unique command interface for VMS and Unix users

Users' Guide

prepared by L.Chiappetti - IFCTR
Dec 1990

Table of content

.pa

0. Purpose

The purpose of the **UNIQ** interface is to define and implement an uniform way by which users of different operating systems may access the more usual commands exactly in the same form, so that they can get accustomed to only one syntax.

This will prevent annoying confusions when one has to remember many different command forms. Morevover it will allow a choice of the command syntax which is more natural and comfortable than the one used by some operating system (although this is somewhat subjective and linked to the preference of the authors).

There are two documents related to the **UNIQ** interface. This is the users guide, which should be sufficient to the general user. A separate definition document collects the requirements and guidelines used in the design of the interface (extensive comments by B.Garilli are gratefully acknowledged), and is available on request.

The guidelines used are the following:

Only the more used, standard "file manager" commands will be affected.

There is almost no attempt to alter the file naming (this is not a programmatic interface of the type provided e.g. by IRAF)

The new commands are implemented as far as possible with plain synonyms (Unix csh aliases or VMS symbols) or by simple procedures (Unix shell scripts or VMS .COM files) at command interpreter level (e.g. DCL or csh). They are not implemented via higher level, compiled language (e.g. Fortran) programs.

The present implementation considers VMS and Unix under csh as target systems. The syntax has been chosen as an intersection (in the authors' taste) of VMS and Unix, to be as close as possible to natural language. MS-DOS was also considered as an example. The above statement means that in general a command should be a pronounciable verb, even if long (DELETE is preferred over rm); and that the presence of "funny" characters (quotes, square brackets, slashes etc.) shall be avoided (unalias is preferred over DELETE/SYMBOL).

.cp50

1. Introduction

This guide is arranged in several sections. After this general introduction, one finds a quick reference section, with various tables, followed by a section with details on specific commands.

The general introduction lists the basic concepts related to the file system as seen by the UNIQ interface : directory (path) names (1.1), file names (1.2), file ownership and protection (1.3).

In the quick reference section (2) table I (section 2.1) lists the functions covered by **UNIQ** commands (plus some other useful commands), and gives the command name, and a reference to the section where one can find more details on the exact syntax (when an explicit reference to section 3 is not given, this is found in Tables II-V).

Tables II and III (for VMS and Unix) in section 2.2 give the system dependent commands equivalent to each UNIQ command. These commands are those which are used by the UNIQ interface to provide its full functions (which are often different from the system native ones, as explained in section 3).

Tables IV and V (again for VMS and Unix) in section 2.3 give instead a correspondence of UNIQ commands with some simple native system dependent commands which provide a reasonable approximation to the wished functions.

Instructions useful for migration to/from other systems and sites, brief notes on the implementation of the command (what to do to implement it on another computer), and also, for commands fully replacing native commands, the instructions on how to disable the UNIQ command can be found in Tables II-V and, in more detail, in section 3.

The complete syntax, and the differences in syntax and performance with the system dependent standard commands, are given in Section 3.

1.1 Directory naming in the UNIQ interface

Directory names are the first part of file names (see 1.2 below). They are different in the two systems :

VMS: device:[dir.dir.dir]
Unix: /dir/dir/dir

Moreover in VMS a logical name can take the place of a device or directory name.

An UNIQ representation is defined for the purpose of some commands only, to be, under UNIQ/VMS, closely resemblant to the (simpler) Unix definition. This applies to the commands which act on directories only (typically CD, QDIR, MKDIR, RMDIR and possibly DIR and RENAME under some circumstances, when no filename is used).

The UNIQ representation of directory names is an Unix (or MS-DOS) like representation, which uses a backslash (\) instead of the Unix slash (/) as separator (slash is a reserved character in VMS). In additions the UNIQ representation recognises some VMS peculiar forms :

UNIQ	VMS equivalent	Unix correspondent	Meaning
.	[]	.	current
..	[-]	..	parent
nothing	SYS$LOGIN	nothing	see notes
\	[000000]	/	root
\dir\dir	[dir.dir]	/dir/dir	absolute
dir\dir	[.dir.dir]	dir/dir	relative
.\dir	[.dir.dir]	./dir	relative
..\dir	[-.dir]	../dir	brother
device:[dir.dir]	device:[dir.dir]	not applicable	as in VMS
device:dir\dir	device:[dir.dir]	not applicable	as previous
[dir.dir]	[dir.dir]	not applicable	as in VMS
device:	device:[000000]	not applicable	
device:\	device:[000000]	not applicable	
logical name	logical name	not applicable	as in VMS
recognised	[...]	not applicable	all subdir
recognised	[*]	not applicable	all one down
recognised	[--]	not applicable	two up
etc.			

Notes:

parent is the directory one level above the current one
brother is a subdirectory of parent

a blank is interpreted in a context dependent way; for CD points to the home (SYS$LOGIN) directory, for other commands to the current directory.

root is the root of the current default device, or of the device explicitly specified

absolute paths refer to root, relative to current directory

any string containing a square bracket is untranslated and assumed to map directly to a VMS pathname

any logical name is immediately recognised and honored appropriately (this may imply a change of device)

anything containing a colon is assumed as a device name. If the device and directory are valid, this is equivalent to issuing a cd device: followed by a cd \dir\dir.

In all cases the UNIQ cd under VMS gives error (and remains where it is) if the target directory is invalid or does not exist.

.cp3

1.2 File naming in the UNIQ interface

Users are reminded that the syntax of a complete file name is quite different on the two systems and is not affected by the UNIQ interface.
In all cases a full file name (not just a directory name, for which see 1.1 above) is required, the system dependent name is used. This name has the form :

VMS: node::disk:[directory]fname.ftype;v
Unix: /filesystem/directory/fname

Note Unix has no explicit file type. Also the wildcard mechanism will be used in system dependent way. Note also the way the different networking/communication protocols and programs (e.g. NFS, ftp, Decnet) map filenames among VMS and Unix systems are not uniform, and often rigid.
Therefore one can just issue some simple recommendations :

Unix users are encouraged to use for their files a single dot in the filename, and to use this consistently with an interpretation of the form fname.ftype (even if this is not usual or necessary in Unix, and is not used by most system files).

VMS users are encouraged to keep a single version of their files, and purge them frequently. For typeless files, it is recommended to add a dot to the end of the filename.

1.3 File ownership and protection in the UNIQ interface

It shall be noted that protection of files is arranged on systems like VMS and Unix with different modalities. In practice this means :

VMS: has four user cathegories (system, owner, group and world, or SOGW)
Unix: has three user cathegories (user, group and other, or ugo)

VMS: has four kind of protections (read, write, execute, delete, or RWED)
Unix: has three kind of protections (read, write, execute, or rwx)

Also the way protections are expressed is system dependent.

The UNIQ interface recognises only two user cathegories :

User maps to VMS OWNER and SYSTEM (they shall have the same protection) and to Unix user.

Other maps to VMS GROUP and WORLD (or Unix group and other). They shall have the same protection (groups are ignored)

The UNIQ interface recognises only three kind of protections. In practice this means that in VMS one shall have both WRITE and DELETE access at the same time (this way it maps to Unix write access).

The UNIQ interface always codes protections as three-letter codes as follows:

rwx Unix rwx, VMS RWED
rw- Unix rw-, VMS RWD

```
r-x       Unix r-x, VMS RE
r--       Unix r,   VMS R
wx       Unix wx,  VMS WED
w-       Unix w,   VMS W
x       Unix x,   VMS E
both no access
```

1.4 Command abbreviation

Command abbreviation is NOT covered by the UNIQ interface, and is left to the operating system native features. In practice this means :

On VMS systems : a native command can generally be abbreviated to the minimum unambiguous abbreviation (or even extension !). This is sometimes done in an excessive way (e.g. if you use a non-existing command PRINTA, VMS will think you are using PRINT !). The degree of abbreviation of an aliased command is determined in its definition (a definition like COM*MAND == value will means that the word COMMAND can be abbreviated up to the three letters COM, i.e. COM, COMM, COMMA, COMMAN). The latter form of abbreviation may be used by the UNIQ interface, and is indicated by the notation **COM**MAND (the part in boldface is compulsory).

On Unix systems : one generally does not have abbreviation enabled. If an abbreviation is wished, this should be explicitly aliased. In the example above one should first define "command" and then explicit aliases for each form like "com", "comm" etc. This is generally NOT done in the UNIQ interface. When abbreviations are used, they are limited explicitly to the particular form indicated : e.g. **his** and **history** but not "hist", "histo" etc.

.cp50

2. Quick reference section

2.1 Command overview

Table I : UNIQ command function overview

Function	UNIQ command	Ref.
logging in	system dependent	3.1
change the user's password	system dependent	-
on-line help	**help** topic	-
recalling the last commands given	**his**tory	-
editing the last command	system dependent	-
changing a file's protection		3.2
user protections	**protuser** prot file	3.2
group and world protections	**protother** prot file	3.2
making a file executable	**makexec** file	3.2
directory listing	**dir** directory	3.3
querying how much disk space		3.4
is available on a disk	**qdisk** disk	3.4
is taken by a directory	**qdir** directory	3.4
is taken by a family of files	**size** file(s)	3.4
creating a new directory	**mkdir** directory	3.5
changing the current directory	**cd** directory	3.6
querying what is the current directory	**pwd**	-
deleting a directory and its content	**rmdir** directory	3.7
copying a file	**copy** file1 file2	3.8
deleting a file	**del**ete file(s)	3.9
renaming a file	**ren**ame file(s) newname	3.10
moving a file to another directory	**move** file(s) newdir	3.10
typing a file to the terminal	**type** file	3.11
looking at a file in full screen	**browse** file	3.11
editing a file	system dependent	-
differences between two files	**diff** file1 file2	note
sorting a file	**sort** file	note
searching a pattern in a file	**search** file pattern	note
querying the system time	**date**	-
querying the status of running programs	**status**	-
stopping a program	**off** process	3.12
compiling and linking a program	see separate document	-

.cp3

printing a file	system dependent	3.13
querying the status of the print queue	**q**ueue	3.13
aborting a print job	**deprint** printjob	3.13
querying the logged-on users	**who**	3.14
identifying who is using a terminal	**id**entify	3.14
sending one-line message to an user	**tell** user message	3.15
chatting with an user	**phone** user	3.15
sending mail to an user	**mail**	3.16
remote file copy	system dependent	3.16
remote login	system dependent	3.16
transferring files via internet	**ftp**	3.17
querying if a remote node is reachable		3.18
via internet	**ping** node	3.18
via decnet	**isup** node	3.18
operating with a magnetic tape	**tape** command	3.19
copying files from/to magnetic tape	to be arranged	
logging out of the system	**log**out	3.20
outputting a text to the terminal	**echo**	-
remove an alias to a command	**unalias**	-

Note: commands diff, sort and search are not strictly part of the UNIQ interface, as they are native commands (except for UNIQ/Unix search which is an alias to the native command grep). Their syntax and usage is heavily system dependent.
.pa

2.2 Equivalence with system commands

2.2.1 VMS equivalences

The next table gives the equivalence of UNIQ commands with VMS commands, considering the implementation adopted. Except for the case an UNIQ command is natively present in VMS (in which case it uses the exact syntax of VMS), all other UNIQ commands are implemented as aliases (technically speaking, VMS global symbols).

In the case they point to an existing VMS command, with or without non-default qualifiers, they are flagged in the third column below as aliases and provide the standard VMS function under a different name. Any argument required by the VMS command is required also by the UNIQ command, with the VMS syntax. The second column gives the equivalent VMS command in

full (if space allows; or a note for otherwise peculiar cases).

All remaining UNIQ commands are implemented as command procedures (all files, whose name is indicated in the third column) reside in directory [LOCAL]). Such commands may be of two types :

providing a variant of a VMS function (e.g. allowing for a different argument syntax, like e.g. the case of the UNIQ directory naming), or provide a function not available in VMS

alter the behaviour of a VMS command, in the case they have the same name (e.g. DELETE, RENAME). In this case pure VMS users shall refer to the detailed explanation to know the difference, and to section 3 for instructions in case they want to revert to the VMS standard behaviour.

Table II : Equivalence of UNIQ command with system commands

UNIQ command	VMS equivalence	Implementation
logging in	return then log in	n/a
change password	use **SET PASSWORD**	n/a
help	**HELP** topic	native
history	**RECALL/ALL**	alias
edit last command	up-arrow key	n/a
change file protection	use **SET FILE/PROT**	
protuser	command procedure	PROTUSER.COM
protother	command procedure	PROTOTHER.COM
makexec	not implemented	not necessary
dir	use **DIR**ECTORY	complex aliasor DIR.COM
.cp3		
query disk space	use **SHOW DEV** or **DIR**	
qdisk	command procedure	QDISK.COM
qdir	command procedure	QDIR.COM
size	command procedure	SIZE.COM
mkdir	command procedure	MKDIR.COM
cd	command procedure	CD.COM
pwd	**SHOW DEFAULT**	alias
rmdir	command procedure	RMDIR.COM
copy	**COPY** file1 file2	native
delete	command procedure	DELETE.COM
rename	command procedure	RENAME.COM

move	command procedure	MOVE.COM
type	**TYPE/PAGE** file	alias
browse	**EDIT/READONLY** file	alias
edit	**EDIT** (for default editor)	native
diff	**DIFF** (system dependent)	native
sort	**SORT** (system dependent)	native
search	**SEARCH** (system dependent)	native
date	**SHOW TIME**	alias
status	**SHOW SYSTEM**	alias
off	**STOP** process	alias
compiling and linking	described in the document "The Real Programmer Tool"	
printing		
default font	use **PRINT** file	native
elite font	use **ELITE** file	complex alias
small font	use **SMALL** file	complex alias
landscape font	use **LANDSCAPE** file	complex alias
queue	**SHOW QUE/ALL SYS$PRINT**	alias
deprint	command procedure	DEPRINT.COM
.cp3		
who	**SHOW USER/INT/BAT/NET/FULL**	alias
identify	command procedure	IDENTIFY.COM
tell	not available	--
phone	**PHONE** user	native
mail	**MAIL**	native
ftp	alias to UCXFTP/ULTRIX	alias
query remote node		
ping	via UCX, system only	alias
isup	command procedure	ISUP.COM
tape	command procedure	TAPE.COM
tape copy	to be arranged	--
logout	**LOG**OUT	native
echo	**WRITE SYS$OUTPUT**	alias
unalias	**DELETE/SYMB/GLOB**	alias

.pa

2.2.2 Unix equivalences

The next table gives the equivalence of UNIQ commands with SunOS Unix commands (using csh), considering the implementation adopted. Except for the case an UNIQ command is natively present in Unix (in which case it uses the exact syntax of Unix), the other UNIQ commands can be implemented either as Unix aliases or by

executable shell scripts.

Items flagged in the third column below as aliases may provide the default Unix function under a different name, or provide an abbreviation of a native command (in this case they are flagged as native/alias). Any argument required by the Unix command is generally accepted by the UNIQ command, with the Unix syntax, but in some cases the purpose of the UNIQ alias is just to alter the order. The second column gives the definition used in the alias command (inclusive of csh placeholders for the arguments or argument list like \!* but without encircling quotes) if space allows; or a note for otherwise peculiar cases.

All remaining UNIQ commands are implemented as csh shell scripts (all files, whose name is indicated in the third column reside in directory /usr/local). Such commands may be of two types :

providing a variant of a Unix function or a function not available in Unix

alter the behaviour of a Unix command, in the case they have the same name (e.g. cd). In this case pure Unix users shall refer to the detailed explanation to know the difference, and to section 3 for instructions in case they want to revert to the Unix standard behaviour.

Table III : Equivalence of UNIQ command with system commands

UNIQ command	Unix equivalence	Implementation
logging in	at prompt log in	n/a
change password	use **passwd**	n/a
help	**man**	alias
history	**history**	native/alias
edit last command	csh !n:s/a/b/ facility	n/a
change file protection	use **chmod**	
protuser	shell script	protuser
protother	shell script	protother
makexec	chmod a+x \!*	alias
dir	ls -aFl \!* | more	alias
.cp4		
query disk space	use **du** or **df**	
qdisk	df \!*	alias

qdir	shell script	qdir
size	shell script	size
mkdir	**mkdir**	native
cd	cd \!* ; prompt setting	complex alias
pwd	echo $cwd	alias
rmdir	**rm -r**	alias
copy	**cp -i**	alias
delete	**rm - i**	alias
rename	shell script	rename
move	shell script	move
type	cat -v \!* | more	alias
browse	**view**	alias
edit	alias to textedit	SunView only
	use xedit if possible	X-window only
	use vi or ed etc.	all other cases
diff	**diff**	native
sort	**sort**	native
search	grep \!:2 \!:1	alias
date	**date**	native
status	ps -acgux \!* | more	alias
off	shell script	off
compiling and	described in the document	
linking	"The Real Programmer Tool"	
printing to Laserjet		
default font	use **print**	alias
elite font	not available yet	
small font	use **small**	alias
landscape font	not available yet	
printing to VAX LN03	use **vprint**	shell script
elite font	default with vprint	
other fonts	specify font to vprint	
queue	lpq \!*	alias
deprint	lprm \!*	alias
.cp3		
who	**who**	native
identify	use echo and environment	complex alias
tell	shell script	tell
phone	**write**	alias
mail	**mail**	native
ftp	**ftp**	native
query remote node		

ping	**ping**	native
isup	alias using dnincp	alias
tape	shell script	tape
tape copy	to be arranged	--
logout	**logout**	native/alias
echo	**echo**	native
unalias	**unalias**	native

.pa

2.3 Correspondence with system commands

2.3.1 VMS correspondences

The next table gives the correspondence of UNIQ commands with the closest VMS commands, considering the implementation adopted. In all cases an UNIQ command is not exactly aliased to a VMS command, this table gives a VMS command which performs similarly, but not identically to the UNIQ command.

This table could be used by someone accustomed to UNIQ commands, when moving to another VMS site, to find out simple VMS commands which do the wished function. If one wants instead to install the entire set of UNIQ commands on another site, one should copy from IFCTR::DUA0:[LOCAL] the file UNIQ.COM (alias definitions), and all other .COM files listed in Table II above, plus any other file eventually required by those .COM files.

The difference in behaviour between the UNIQ commands and the VMS correspondences listed below are given in section 3.

Table IV : Closest system commands corresponding to UNIQ commands

UNIQ command	VMS equivalence
logging in	standard login procedure
change password	**SET PASSWORD**
help	**HELP** topic
history	**RECALL/ALL**
edit last command	up-arrow key or **RECALL** n
change file protection	use **SET FILE/PROT** in one of its variants
protuser	**SET FILE/PROT(O**:prot,**S**:prot**)**
protother	**SET FILE/PROT(G**:prot,**W**:prot**)**

makexec	function not required in VMS
dir	**DIR**ECTORY with optional qualifiers
query disk space	use closest match as indicated below
qdisk	**SHOW DEV** device
qdir	**DIR/GRAND/TOTAL** directory
size	**DIR/GRAND/TOTAL** files
mkdir	**CREATE/DIRECTORY** directory
cd	**SET DEFAULT** directory
pwd	**SHOW DEFAULT**
rmdir	cannot be done in a simple way
.cp4	
copy	**COPY** file1 file2
delete	**DELETE** filename.typ;v
rename	**RENAME** file1 file2
move	**RENAME** file1 file2
type	**TYPE** file (default is /NOPAGE)
browse	**EDIT/READONLY** file
edit	**EDIT** file (for default editor)
diff	**DIFF** (see on-line help)
sort	**SORT** (see on line help)
search	**SEARCH** (see on line help)
date	**SHOW TIME**
status	**SHOW SYSTEM**
off	**STOP** process
compiling	**FOR**TRAN
linking	**LINK**
printing	**PRINT** file
queue	**SHOW QUEUE**
deprint	**DELETE/ENTRY** printjob
,cp3	
who	**SHOW USER**
identify	**SHOW PROCESS**
tell	**REPLY** (operators only)
phone	**PHONE** user
mail	**MAIL**
ftp	**FTP** (uses VMS syntax, not Unix like)
query remote node	
ping	allowed with system privileges only via UCX
isup	**NCP TELL** node **SHOW EXEC**

tape	use **SET MAGTAPE**
tape copy	
logout	**LOG**OUT
echo	**WRITE SYS$OUTPUT** string
unalias	**DELETE/SYMB/GLOB** symbol

.pa

2.3.2 Unix correspondences

The next table gives the correspondence of UNIQ commands with the closest Unix commands, considering the implementation adopted. In all cases an UNIQ command is not exactly aliased to a Unix command, this table gives a Unix command which performs similarly, but not identically to the UNIQ command.

This table could be used by someone accustomed to UNIQ commands, when moving to another Unix site, to find out simple Unix commands which do the wished function. If one wants instead to install the entire set of UNIQ commands on another site, one should copy from sun!/usr/local the shell script file uniq (alias definitions), and all other shell scripts listed in Table III above, plus any other file eventually required by those files.

The difference in behaviour between the UNIQ commands and the Uniq correspondences listed below are given in section 3.

Table V : Closest system commands corresponding to UNIQ commands

UNIQ command	Unix equivalence
logging in	local login procedure
change password	**passwd**
help	**man** topic
history	**history** n
edit last command	a general, but impractical way is provided by csh. Otherwise, in SunView, use the cut and paste facility.
change file protection	use **chmod** with all its variants
protuser	(for details see man page for chmod)
protother	(for details see man page for chmod)
makexec	**chmod a+x** file
dir	use **ls** with all its variants

query disk space	use **du** or **df** or perhaps **find**
qdisk	**df** filesystem
qdir	**du -s** directory
size	**du -a** file(s)
mkdir	**mkdir** directory
cd	**cd** dir (does not set the prompt by default)
pwd	**pwd**
rmdir	**rmdir** dir (not recursive or unconditional)
.cp4	
copy	**cp** (may be not protected for overwriting)
delete	**rm** (may be not protected for overwriting)
rename	**mv** (may be not protected for overwriting)
move	**mv** (may be not protected for overwriting)
type	use **cat, more** or **page**
browse	**view**
edit	use textedit in SunView, xedit in X-window and vi or ed etc. in all other cases
diff	**diff**
sort	**sort**
search	use **grep** or similar commands
date	**date**
status	use **ps** with all its variants
off	use **kill -9** with process number
compiling and linking	**f77** generally handled by **f77** (else use **ld**)
printing	the default Unix command is **lpr** a Laserjet attached to the tty port works better if some escape sequences are sent together with the print job by an appropriate alias a local command **vprint** allows dispatching the printouts to the VAX LN03
queue	**lpq**
deprint	**lprm**
.cp3	
who	**who**
identify	**who am i**
tell	use **write**
phone	**write**
mail	**mail**
ftp	**ftp**

query remote node

ping	**ping**
isup	**dnincp show node** node

tape	use **mt**
tape copy	
logout	**logout** (executes .logout file)
echo	**echo**
unalias	**unalias**

.cp50

3. Full reference section

3.1. Logging in

$ELISMThis is not strictly part of the UNIQ interface$ELITE

The user will automatically find most of the necessary definitions set up for him at login by the systemwide login procedure. This procedure is run before the user's private login procedure, and is, in our local implementation :

VMS: the SYLOGIN.COM procedure in the system area
Unix: the .login and .cshrc in the user home area

The user can arrange a personal, customized login procedure (in Unix this is a procedure executed in each shell, not only in the login shell) by creating the following file in the home directory.

VMS: LOGIN.COM
Unix: .mycshrc

Note for Unix users : this is unlike normal Unix usage, and for this reason .login and .cshrc are not real files, but links to a common version in the system area.

The UNIQ prompt indicates the current directory, and is automatically updated when one changes directory using an UNIQ command, similarly to what happens in MS-DOS. The prompt is set as follows :

VMS: the last two components of the current path
Unix: the last component of the current path and the command number

3.2 Change file protection

$ELISMThe next paragraph is not strictly part of the UNIQ interface$ELITE

Default file protections (applied to all newly created files) are normally assigned in the login procedure. If an user wants to change the default protection, one should use the following system dependent commands. Note that the syntax and logics of the two commands is radically different (Unix uses numeric masks); see the relevant help files :

VMS: SET PROT/DEFAULT
Unix: umask

The UNIQ commands to assign file protections other than default are the following :

protuser prot file
protother prot file
makexec file (Unix only)

The first command assigns protections to the user cathegory (for VMS this is equal to OWNER and SYSTEM at the same time). The second one assigns protections (identical) to the cathegories of group and other (or GROUP and WORLD). A protection is expressed in UNIQ syntax as a three letter code : rwx, r-x, etc. The three letters are (in order) r,w, or x, (to set read, write-and-delete or execute bits), or a - (to remove the respective protection). It is compulsory to specify all three values at the same time.

The third command is needed on Unix files only, and makes a file executable (assigns execute access to user, group and world).

3.3 Directory listing

The UNIQ/VMS DIR command overwrites the default VMS command, which remains available if called as **DIRE**CTORY (4 or more letters).

The UNIQ interface makes a choice for the quantity and format of information displayed for each file. This is customizable as follows:

If the VMS user does not like the format used to display the

listing, one shall not redefine the DIR command, but either **unalias MYDIR** to obtain VMS default short listings, or redefine symbol MYDIR with the wished VMS DIRECTORY command. The one used by the UNIQ interface is :

```
DIRECTORY/SECUR/DATE=MOD/SIZE=ALL/WIDTH=
(DISP=80,OWN=12,FILE=31)
```

If the Unix user does not like the format used to display the listing, one can realias **dir** to a different combination of ls flags (see the manual page for ls). It is recommended to pipe the output thru **more**.

3.4 Disk space available

The UNIQ commands **qdisk**, **qdir** and **size** do not overwrite any system command (with the exception of **size** under Unix, which makes unavailable an obscure and seldom used command). The syntax and parameters of the three commands are :

qdisk disk

where in VMS disk is a valid physical device, and in Unix is a filesystem or directory specification (in the latter case the information is returned for the entire filesystem to which it belongs).

returns the total disk space occupied and free (also as percentage of the total) for a VMS physical disk or an Unix filesystem.

qdir pathname

where in VMS pathname is a VMS filemask or UNIQ directory specification, and in Unix a directory specification. In VMS this returns the space taken by the directory without descending in subdirectories (unless one explicitly uses a form like **qdir [**dir...**]**). In Unix it always descends in the subdirectories.

.cp2

size file(s)

where file(s) is a (system dependent) filename with wildcards. Returns the total space occupied by a family of files with some common name. In Unix ignores directories and links.

All the above commands return the quantities in kbytes (VMS blocks are 0.5 kbytes).

3.5 Create a new directory

The UNIQ command **mkdir** directory in VMS accepts directory specification in VMS and UNIQ format. It is aliased to CREATE/DIRECTORY/VERSION=5.

3.6 Changing current directory

The UNIQ command **cd** directory in VMS accepts directory specification in VMS and UNIQ format.
In both cases of VMS and Unix the cd command also updates the system prompt. If a VMS user wants to change the default prompt he should **unalias PROMPT** (to have the default VMS prompt) or redefine symbol **PROMPT** to the SET PROMPT he wishes (to have a fixed prompt) or to a procedure of his choice (for a variable prompt). If an Unix user wants to change the default prompt, he should realias the entire cd command.

The UNIQ default prompt indicates the current directory, and is set as follows :

VMS: the last two components of the current path
Unix: the last component of the current path and the command number

3.7 Deleting a directory

The UNIQ command **rmdir** directory deletes a directory unconditionally (i.e. even if not empty removes all files in it) and recursively (i.e. removes all subdirectories). In VMS one can use VMS or UNIQ directory names.

Unix users may restore the default behaviour or rmdir (conditional to the directory being empty) by unaliasing rmdir.

In VMS rmdir is aliased to a complex procedure which replaces the usual VMS lengthy manual equivalent. Such procedure implies also changing protection to the .DIR file itself (normally it is delete-protected) and to any delete-protected files.

3.8 Copying files

The UNIQ command **COPY** (in both VMS and Unix) forbids copying over an existing file.

.cp3

3.9 Deleting files

The UNIQ command **DELETE** file has a different behaviour in VMS and Unix.

In Unix it will ask permission for deletion. This is useful due to the "typeless" (and confusing) characteristic of Unix files, to prevent unwanted deletions.

In VMS this command is a procedure, which makes unavailable the normal VMS DELETE commands (inclusive of DELETE/ENTRY, etc.). The procedure will automatically delete all versions of a file if no explicit ;n or ;* is specified. To pass file DELETE qualifiers (e.g. /CONFIRM) they shall follow the file argument and be separated by a blank.

To access standard DELETE (and DELETE/ENTRY etc.) use the alias **VMSDELETE**. This works as the normal VMS command.

VMS users wishing to revert to the normal behaviour, just unalias DELETE.

.cp5

3.10 Renaming and moving files

The UNIQ command **RENAME** oldname newname behaves differently than the equivalent VMS (RENAME) or Unix (mv) command. In both systems it allows renaming only keeping the renamed file in the same directory.

The move-to-another-directory function is performed by a separate UNIQ command **MOVE** file newdir, which moves the file to a different directory with the same filename (note that moving is much more efficient than copying and deleting). On UNIQ/VMS newdir can be expressed in VMS or UNIQ directory naming.

The default system function is available to VMS users as an alias **VMSRENAME**, and to Unix users as the native mv command. VMS users may revert to the default behaviour by unaliasing RENAME.

Both UNIQ commands operate also with wildcards.

The UNIQ/Unix rename command implements file names and types (the file type is defined as the suffix after the last dot, the file name is all the rest). The filename and filetype portions of newname can be specified as **=** to have it copied from the corresponding part of oldname.

The UNIQ/VMS RENAME commands acts differenty on file names and types. If the filetype (inclusive of the dot) is not specified (the dot is not present) in newname it defaults to the same filetype as in oldname.Similarly if the filename is absent (nothing before the dot), it defaults to the same as in oldname.

The following examples work in the same way :

```
rename pinco.panco =.pallino  in UNIQ/Unix and
RENAME PINCO.PANCO  .PALLINO  in UNIQ/VMS  or

rename tizio.caio  giulio.=   in UNIQ/Unix and
RENAME TIZIO.CAIO  GIULIO     in UNIQ/VMS
```

give respectively a new name of pinco.pallino and giulio.caio.

3.11 Typing a file

The UNIQ command **TYPE** is defined to stop at the end of each page. Unix users should be aware they are using all the facilities of the **more** command. To revert to non-paginated typing, Unix users may use cat, and VMS users can just unalias TYPE (or use TYPE/NOPAGE if this is required for a single run).

The BROWSE command allows to access the file in a full screen edit mode (using EDT in VMS and vi in Unix), but with the file being protected against accidental changes.

We remind here of the VMS **DUMP** and Unix **od** commands, useful to look a binary files in hexadecimal, octal, etc.

3.12 stopping a program

The behaviour of the UNIQ command **OFF** progname is quite different due to instrinsic differences between VMS and Unix, and they are also quite unlike what one expects from a well-behaved operating systems (e.g the old HP RTE-6).

In VMS every program usually runs under a single process. To terminate the current program (or image like they call it) one uses a ctl-Y. In the rare case one has subprocesses concurrently active (e.g. with IRAF or SMONGO, or using SPAWN), one can very simply stop each process by name (process names are given by **STATUS** or **WHO**). This uses the plain VMS STOP command (this works for processes owned by the user; this means the UNIQ command OFF wont' work for SYSTEM).

In Unix each command runs in a process (or even more than one !). The UNIQ command therefore looks for processes with a given name, finds the relevant number (or pid : process id) and kills it. If there are more processes owned by the user, one is presented with the list, and should repeat the OFF command using the pid as progname. In the case the user is superuser, the UNIQ OFF command displays also processed owned by others.

3.13 Printing

As printing may occur in a lot of different ways, this is not handled completely by the UNIQ interface.

On VAX the VMS **PRINT** command (used by numerous other utilities) is left alone (this will normally print in the default Courier font). Additional commands are added to handle printing in elite, small and landscape fonts, and take the filename as argument, eg. **ELITE** file.

.cp5
As our Sun has no permanent alphanumeric printer attached, the question is not relevant. A **print** alias is defined with implicit escape sequences for the Laserjet (when attached). The normal way to print files is to use the **vprint** file font command to send prints to the VAX LN03. If font is not specified it defaults to **elite**.

> To stop a print job with the UNIQ command **DEPRINT** jobnumber, one shall obtain the job number with the UNIQ command **QUEUE.**

Note: VAX print jobs sent by vprint can be killed only by logging on the VAX as user SUNPRINT and selecting menu option 3.

3.14 Logged on users

The UNIQ commands **WHO** and **IDENTIFY** have no parameters. Their output should be self-explanatory. Note that the UNIQ/VMS WHO displays also non-interactive users.

IDENTIFY is useful to know who is using that free terminal to which you are lounging.

3.15 Messages to users

The UNIQ command **TELL** (so far defined only for Unix) is a simple way of sending short messages. This is preferred to the use of VMS PHONE or the equivalent Unix talk, which require the entire screen and force the user to stop what is doing.

3.16 Mail, file exchange and remote login

This is described in a separate document "The Link Guide : an overview of networking within IFCTR computers".

3.17 ftp file exchange

The UNIQ arrangement for FTP under VMS is to use the Unix-like interface available under UCX, in preference to the VMS one. Users wishing to revert to the VMS interface just unalias FTP. Note however that two aliases **FTPV** and **FTPU** will be permanently defined (not by @UNIQ, but by @LOCAL procedure) to allow use of respectively the VMS and Unix interfaces.

3.18 inquiring on network

For TCP/IP connections the only way to know whether a node is there is the Unix **ping** command. There is no way to know what is the path to reach it and so on. The equivalent of ping under VMS UCX is accessible to SYSTEM only.

For Decnet connections on VMS systems the SHOW NET command is quite painful. Usage of **NCP** commands is better, however finding the exact path to a node requires lot of interactions with NCP (similarly to the RSCS SMSG in Bitnet). The UNIQ **ISUP** commands provides NCP information on a remote node.

Note that the implementation of ISUP is different : on VMS it

asks the remote node to identify oneself, while on Sun it queries the local data base to check whether the node is there. The first way may be slightly slower, but always works (unfortunately is not implemented by Sun dnincp), while the second often returns incorrect information (this is felt unimportant for the Sun, where one is interested only to local connections, which should be OK).

3.19 Handling tapes

The purpose of the **TAPE** command will be that of selecting a tape drive (according to the system, this could include the allocation of the drive), querying its status, setting tape characteristics (like density), skipping files and blocks and writing tape marks. This will be implemented by a command like :

tape use tapename

to select a particular tape drive (tapename will be system dependent), and by a command like :

tape tapecommand parameter(s)

to issue a given command to the selected tape drive.

This command is presently not implemented and will be described in a separate "guide to use of tapes" document (to be written)

3.20 Logging out

$ELISMThis is not strictly part of the UNIQ interface$ELITE

The local implementation of the **log**out command is as follows :

For Unix : **log** is aliased to the standard logout command. The .logout file is executed while logging off. In the local implementation this file is a link to a systemwide logout procedure. This procedure includes some disk cleanup (removal of unwanted files), and a call to the "fortune" command.

For VMS : the **LO**GOUT command is aliased to a procedure LOGOUT.COM, in order to somehow emulate the Unix behaviour. This procedure (located in [LOCAL]) does what follows:

first finds if a private LOGOUT.COM file exists in the user's login directory (if yes it executes the commands in there), then it executes the systemwide procedure in [LOCAL.SYLOGIN] SYLOGOUT.COM (which does some disk cleanup and prints a farewell message), finally it unaliases itself and logs out.

VMS users are warned NOT to include a LOGOUT command in their private LOGOUT.COM. If they will do it, an infinite loop may occur.