Introduction to Unix and VNMR

created 3/5/95 – updated 12/19/01

I. Common Unix Commands

cd ~	– change directory to home (dir you first go to on login)
cd ~/vnmrsys	– change to your vnmr directory
pwd	– show current location (path)
df -k	– display file system (will show free space)
☞ ***paths***	– see section III on directory structures
mkdir	– make directory
rmdir	– remove directory
man *cmnd*	– show manual pages for *cmnd*
ls	– list files
ls -la	– list files with options lga
alias dir 'ls -la'	– in your .cshrc file
cp -r *name*	– copy recursively from *name* down path (i.e., including all subdirectories)
mv *fname tname*	– move *fname* to *tname* (i.e., a rename unless a change in path is specified)
rm -r *name*	– remove from *name* down path (including all subdirectories) use care; this is a dangerous command!

Use the text editor in the CDE windowed environment if possible (much easier than vi), but always finish with a carriage return at the end of the file when creating macros. In addition, you can use *textedit* *filename* at a unix prompt to bring up the more-usable textedit editor.

vi *filename*	– edit *filename* using vi editor (*cgf* has couple pages of Vi Quick Reference)
i	– insert (must use **ESC** or double up/down arrow to exit insert mode)
a	– append; insert *after* cursor position (use at end of line)
x	– delete single character (**10x** will delete 10 characters)
dd	– delete single line
^	– move to beginning of line (then use **i** to insert at beginning of line)
$	– move to end of line (then use **a** to insert at end of line)
:wq	– write and end vi session
:q!	– quit, discard changes
***/text*<ret>**	– go to next occurrance of *text*
:set number	– gives line numbers, help with debugging

II. Common VNMR Commands, Parameters and Flags

(☞ particularly useful commands)

☞ **dg**	– display group (normal parameters)
dg1	– display second (processing, plot... parameters)
dgs	– display shim group
☞ **dps**	– display pulse sequence
da	– display array
☞ **array**	– powerful array command, for stacked acquisition (kinetic exps, pw checks)
☞ **full**	– uses full screen; needed after **dssh** and other commands (is NOT the same as the DISPLAY INTERACTIVE FULL button)
wysiwyg='n'	– sets display so scaling does NOT match printer
svf('*filename*')	– save file (1d or 2d) with associated data, phasing, etc. (FILE SAVEFID)
rtf('*filename*')	– reads file (menu alternative MAIN MENU FILE)
svp('*filename*')	– save only parameters (1d or 2d); significant savings in disk space!!
rtp('*filename*')	– reads parameters
svs('*filename*')	– save shim settings to user directory ~/vnmrsys/shims (FILE SAVESHIM)
rts('*filename*')	– reads shims (follow with **loadshims** macro; UWMACROS LOADSHIMS)
wexp='svf(\'*filename*\')'	– saves data to *filename* at completion of experiment (\' required for quotes within the main quotes – requires **au** used to start acquisition
explib	– lists all experiment areas (menu MAIN WORKSPACE LIBRARY)
cexp(*#*)	– creates experiment area number # (see also WORKSPACE)
delexp(*#*)	– deletes experiment area number # (save disk space; see also WORKSPACE)
temp24	– UW macro to set temp to 24°C (cp into your ~\vnmrsys\maclib, rename and edit for other temperatures); (UWMACROS SET TEMP)
vttype=0 or **2**	– variable temp controller not present (=0) or present (=2)
temp='n'	– no temp specificied for control
vttype=0 temp='n' su	– command preceeding probe change (see macro *tempoff*)!!
vttype=2 temp=24 su	– command to re-establish temp control at 24°C *after* probe change (see macro *temp24*)
tn='Si29'	– set transmitter to ^{29}Si frequency (uses look-up table; see macro *tuneh*)
d# (e.g., d1=1)	– set delay time in seconds (use dps to make sure about timing)
pw=8	– common pulse width in µsec (use dps)
p# (e.g., p2=4000)	– pulse width in µsec (use dps)
tpwr	– observe transmitter power set in dB, from 0 to 63, larger is more power
tpwrf	– observe transmitter fine power control, from 0 to 4095, ~3db spread
nl	– first set cursor close to peak, then finds nearest line
☞ **rl(-9.86p)**	– first use nl to set cursor exactly on, then sets reference
rl1(-9.86p)	– expand about correct reference on diagonal peak, use to ref *f1* axis
lb=5	– gives exponential broadening of 5 hertz with **wft** or **ga**

wft	– weighted (depending on parameters set) FT
☞ **aph**	– good automatic phase to at least start phasing
np=4k	– often set by *sw*, this directly sets to 4096 points acquired, changes *aq*
fn=16k	– number FT to, with previous command does two zero fills
lsfid=2	– left shift two points before FT (increase if getting baseline rolling)
☞ **movesw**	– macro sets sweep width *sw* to right and left cursor positions
cz	– zero all integral resets (use menu button **reset** and mouse for resets)
bc	– baseline correction with default spline fit; depends on integral resets
bc(5)	– baseline correction using 5th order polynomial fit
pl	– plot spectra(um) as displayed
text('*text goes here*')	– text header plotted on top of page
atext('*more text*')	– adds lines to text header
pap	– plot all parameters
ppa	– plot only a few parameters
axis='p'	– set axis to ppm ('h' for hertz, 'k' for kilohertz)
pscale(-3)	– plot scale (in current units)
axish pscale(-6)	– macro to set Hz scale and plot axis 6 cm below spectrum
page	– go to next page for plotting
clradd spadd	– clear exp5 and place current spectrum in exp5 (setup for addi)
addi	– add/sub (dual display-like) routine
s1	– save display parameters (can use s1 through s9)
r1	– recall display 1 and display as previously saved with **s1**
md(3,4)	– move saved display regions (s1 to s9) from exp3 to exp4
dssh	– display stack horizontally
full	– return display of one spectrum to full width
dss	– display stack using **v0** and **h0** as vertical and horizontal offsets

☞ UWChemMRF *macros*

tuneh	– tn='H1' gain=0 su .... (use UWMACROS TUNE H1)
tunesi	– tn='Si29' gain=0 su....................................(use UWMACROS TUNE SI29)
tunec	– tn='C13' gain=0 su
loadshims	– load='y' su load='n'..(UWMACROS LOADSHIMS)
axisp	– axis='p'
axish	– axis='h'
dsx	– wft dscale(-3)
invert	– rp=rp+180 inverts the 0-order phase of a spectrum
tempoff	– vttype=0 temp='n' su ....crucial command before probe change

	temp24	– vttype=2 temp=24 su vttype=0 su................................model temp macro
	settemp	– vttype=2 temp=x su [wait] vttype=0 su
☞	**intnorm**	– normalizes peak to user specified value (e.g., methyl peak to 3)
☞	**disp2d**	– sets up **wp sc** nicely for 2d plots
	pconpos	– plots contours of 2D spectra with positive peaks getting 10 contours, negative peaks getting just 1
	pconneg	– opposite of pconpos
	lpforward	– sets up for forward linear prediction

III. VNMR Directory Structure

There is a user *vnmrsys* area and a system *vnmr* area. All files are looked for in user area first; if not found there, then the system area is checked.

cd ~	move to home account for computer working at /export/home/*username*
cd /vorlon/fry	moves to /export/home/fry on vorlon workstation
rtf('*datafile*')	reads data (fid) file; *datafile* name must have .fid as suffix (vnmr always appends as such with svf command) all *datafile*.fid files are actually directories containing: fid .... actual fid data log .... log file shows times for acquisitions procpar .... processing parameters text .... a text file created by command text('...')
rtp('*parfile*')	reads parameter file; *parfile* name must have .par as suffix (vnmr always appends as such with svp command) all *parfile*.par files have only procpar and text files

~/vnmrsys

/exp1	data acquisition area; can go up to exp9, but can take significant disk space
/maclib	user macro library
/manual	user manual (help for pulse sequences) directory
/menulib	user menu definitions directory
/psglib	user created or modified pulse sequences
/seqlib	user compiled pulse sequences
/shapelib	user created RF and gradient shapes (e.g., for NOESY1D)
/shims	user stored shim files

/vnmr

/maclib	system macro library
/manual	system manual (help for pulse sequences) directory
/menulib	system menu definitions directory
/psglib	system pulse sequences
/seqlib	system compiled pulse sequences
/shapelib	system stored RF and gradient shapes (e.g., for NOESY1D)
/shims	system (facility) stored shim files

IV. Common Pulse Sequence Setup in VNMR

[this section is directed at students wanting to know details of the vnmr psg setups]

1. Always start by reading in standard parameters using MAINMENU SETUP

2. Then continue by running a macro (e.g., **cosy** runs the cosy macro in /vnmr/maclib) or use SETUP SEQUENCE COSY (reads in the same macro).
3. The better macros read in *only* the parameters specified in **psgset** macro statements within the primary macro routine (only ProteinPack macro do not, and use **rtp** instead). Typically the parameter file found in /vnmr/parlib has the same name as the macro, but not always:

   e.g. macro **hmqc** has statement **psgset('hmqc13c',dg,dg1,pwx,pwxlvl,dpwr,...)**

   which reads the specified parameters in from /vnmr/parlib/hmqc13c. Reading in **dg** and **dg1** setup the appearance of the parameter display in the **dg** and **dg1** screens.
4. The macros for 2d sequences call another macro **set2d** that sets up **np fn ni fn1** based on the desired digital resolution: e.g., **set2d('dqcosy',3,6)** sets up 3 Hz/pt in F2 and 6 Hz/pt in F1.
5. Usually, but not always **set2d** sets the actual pulse sequence used: e.g., inside **set2d** is a statment **seqfil=$1** where $1 is the first parameter passed into the macro (i.e. in this case, the sequence name). **seqfil** is the compiled pulse sequence (psg) residing in /vnmr/seqlib.
6. The pulse sequence source C code resides in /vnmr/psglib. The source code is compiled with the statement **seqgen** (e.g., for cosy **seqgen relayh.c**). Only users and vnmr1 can compile source code, so common code must be placed in the user's vnmrsys/psglib and compiled. The compiled code is automatically placed into the user's vnmrsys/seqlib which must then be copied into /vnmr/seqlib and have the owner set to vnmr1:nmr (e.g., **chown vnmr1:nmr /vnmr/seqlib/relayh** ; root permission is required for the copy and chown unless the user is vnmr1.
7. When writing or editing a pulse sequence, it is easiest to begin with a premade parameter set. New parameters must be created with the **create(parameter<,type<,tree>>)** command. See section 5.2 in the Pulse Programming manual. It is important to correct define the parameter when creating it: e.g., when making a new variable **dpw2d** for a decoupler power setting that should not go above 35 to prevent probe damage, use the command **create('dpw2d','integer')** and follow that with **setlimit('dpw2d',35,0,1)**. Note that pulse length parameters should be defined with **create('pwsl','pulse')** to insure the units for the pulse are in µs. See the **create** command information in the vnmr documentation for other parameter types.
8. Some newer sequences being written at Varian usually now use a phase table named the same as the sequence and located in /vnmr/tablib, although the newest sequences avoid the use of tablib by specifying the table within the pulse sequence itself (rather than externally).
9. Typically the macro finishes by setting some parameters directly, and then displaying the help file for the sequence that resides in /vnmr/manual via the command **man('relayh')** as and example for cosy.
10. Keep in mind that all vnmr usage looks for macros, parameters, pulse sequences, etc. in the userlib (~/vnmrsys) first, and uses that copy if it exists, prior to looking in the system area (/vnmr). Thus, every user can have their own version (not recommended!) of the same sequence. User's are encouraged to copy a sequence into a new name in their own directory. Finally, I am beginning to implement an intermediate area for facility-developed software, e.g., **maclibpath='/vnmr/maclib.path'** . VNMR will look first at ~/vnmrsys, then at /vnmr/maclib.path, and last /vnmr for a macro. I will update documentation here when this implmentation is done.