NIRSPEC

UCLA Astrophysics Program **U.C. Berkeley** **W.M. Keck Observatory**

James Larkin **Mar 20, 1999**

NIRSPEC Software Programming Note 23.00
Observing Scripts

1. General Description

This script document summarizes the csh scripts generated by the echelle format simulator (see NSPN 14.00) and discusses how a user can generate personalized scripts for themselves. Scripts serve two purposes: first they provide a way for the user to set up observation sequences ahead of time, second they provide a communications link between the EFS and the NIRSPEC server.

Scripting for NIRSPEC is done in csh which is the standard UNIX command language. Each csh script is simply an ASCII file with a series of UNIX commands. This document should be sufficient for preparing simple scripts and for understanding the EFS scripts, but knowledge of csh scripting will allow an advanced user to create much more elaborate scripts.

2. Basic Commands and Uses

There are two fundamental commands that are used to control the NIRSPEC instrument. These are **show** and **modify** and they are the standard control words for all KTL based instrument systems at the Keck Telescopes. Both commands are small c programs that use the KTL library to connect to a server control system (like NIRSPEC's server) and then either modify one of the keywords (**modify**), or retrieve its value (**show**). Since the server controls everything with keywords, being able to modify them gives the user complete control of the instrument. A complete list of keywords is given in KSPN XX.00. For most applications, the syntax for using these commands is:

modify -s *server_name keyword=value*
show -s *server_name keyword*

Some examples are:

```
modify -s nirspec itime=10.0
```
This example sets the exposure time for 1024 frames to 10.0 seconds.

```
show -s nirspec coadds
```
This example reads the value of the coadds keyword that sets how many coadds to use with the 1024 array.

```
show -s dcs2 ra
```
This example reads the *ra* keyword from the telescope control system for the Keck II telescope.

```
modify -s nirspec go=1
```

This example sets the go keyword to 1. This keyword controls taking exposures with the 1024 array. By showing its value, you can check to see if an exposure is currently in progress.

```
set coadd = `show -s nirspec coadds`
set coadd = $coadd[3]
```

This example uses a **show** command to define a csh variable called coadd. The raw version of **show** returns a small reply like "coadds = 2". To put into a variable, the second line of the example is needed to cut out the third element of the string and stick it into coadd. To use this variable later, just put a dollar sign before it like:

```
modify -s nirspec coadds=$coadd
```

This is a simple way of remembering and resetting parameters within script.

As you might have guessed, simple scripts can be created simply by stringing together shows and modifies. An example would be:

```
modify -s nirspec comment='Starting script'
modify -s nirspec itime=10.0
modify -s nirspec coadds=2
modify -s nirspec go=1
modify -s nirspec comment='Script is completed'
```

This simple script sets the integration time, and number of coadds and then starts an exposure with the 1024 array. In most cases, SCAM keywords are identicle to the 1024 except that a '2' follows each keyword such as: itime2, coadds2, and go2.

The **comment** keyword is used to add comments to the FITS header, but only the latest value is used in the next exposure, so you can modify this keyword many times and only the last one goes into the header. But the current value is displayed in the dataviews window so this is a way of sending multiple messages to the user. A good script should be peppered with **modifies** of the **comment** keyword in order to let the user know where the script is in its execution.

3. NIRSPEC startup and configuration scripts

The most important script is the NIRSPEC configuration script:
/kroot/kss/nirspec/keyword/NirspecConfig.csh
It is sourced by the server program 5 seconds after startup, and it can be used to set any initial conditions. Currently it sets the quadrant offsets for both arrays, the detector bias, and the temperature recording times.

There are also a series of scripts that run up the different parts of the NIRSPEC software. These are detailed in NSPN 10.02: Program Launching. Here we just list them:

```
/kroot/kss/nirspec/keyword/run_dcs
```

```
/kroot/kss/nirspec/keyword/run_nirspec_server
/kroot/kss/nirspec/keyword/stop_dcs
/kroot/kss/nirspec/keyword/stop_client
/kroot/kss/nirspec/keyword/stop_nirspec
/kroot/kss/nirspec/keyword/stop_nirspecsim
/kroot/kss/nirspec/ui/xnirspec/run_client
/kroot/kss/nirspec/ui/xnirspec/run_dql
/kroot/kss/nirspec/ui/xnirspec/run_eaves
/kroot/kss/nirspec/ui/xnirspec/run_efs
/kroot/kss/nirspec/ui/xnirspec/run_imrot
/kroot/kss/nirspec/ui/xnirspec/run_nirspec_client
/kroot/kss/nirspec/ui/xnirspec/run_nod
/kroot/kss/nirspec/ui/xnirspec/run_ql
/kroot/kss/nirspec/ui/xnirspec/run_rql
/kroot/kss/nirspec/ui/xnirspec/run_sefs
/kroot/kss/nirspec/ui/xnirspec/run_sql
/kroot/kss/nirspec/ui/xnirspec/run_temp
/kroot/kss/nirspec/ui/watch_imrot/specific/nirspec/startrot
/kroot/kss/nirspec/ui/watch_imrot/specific/nirspec/stoprot
/kroot/kss/nirspec/ui/watch_imrot/specific/nirspec/irot_wrapper
```

4. NIRSPEC Helper Commands

A simple way to call scripts from within other scripts is with the **source** command. This also works at the command line where you can type "source script.csh" to execute a script named **script.csh**. A script can also **source** another script and so we can start creating scripts that are like subroutines or functions. For the EFS scripts and for general scripting we have created a set of scripts (and executables in other languages such as TK) that can help with NIRSPEC control.

For each NIRSPEC mechanism, or exposure type, we have created a script that waits for it to complete. These are all essential to make sure all moves are complete before starting a new exposure, and vice-versus. They are all located in "/kroot/kss/nirspec/ui/csh" and have names starting with "wf". They are all symbolically linked into the standard path /kroot/rel/default/bin so they are callable from any location simply by typing their names. If they fail, meaning a motor doesn't go to "ok" status, or an exposure doesn't complete on time, then they use the ps command to determine the process id of their parent (usually the script that called them). This process is then killed and the script keyword is set back to 0. Finally, upon failure a TK script is started that pops up a window with the failure message. There is a TK script for each WF script. Currently the helper scripts are:

```
/kroot/kss/nirspec/ui/csh/wfcalc
/kroot/kss/nirspec/ui/csh/wfcalc.tk
/kroot/kss/nirspec/ui/csh/wfcalm
/kroot/kss/nirspec/ui/csh/wfcalm.tk
/kroot/kss/nirspec/ui/csh/wfcalp
/kroot/kss/nirspec/ui/csh/wfcalp.tk
/kroot/kss/nirspec/ui/csh/wfdisp
/kroot/kss/nirspec/ui/csh/wfdisp.tk
/kroot/kss/nirspec/ui/csh/wfechl
/kroot/kss/nirspec/ui/csh/wfechl.tk
/kroot/kss/nirspec/ui/csh/wff1
/kroot/kss/nirspec/ui/csh/wff1.tk
/kroot/kss/nirspec/ui/csh/wff2
/kroot/kss/nirspec/ui/csh/wff2.tk
/kroot/kss/nirspec/ui/csh/wffilter
```

```
/kroot/kss/nirspec/ui/csh/wffilter.tk
/kroot/kss/nirspec/ui/csh/wfg
/kroot/kss/nirspec/ui/csh/wfg.tk
/kroot/kss/nirspec/ui/csh/wfg2
/kroot/kss/nirspec/ui/csh/wfg2.tk
/kroot/kss/nirspec/ui/csh/wfirot
/kroot/kss/nirspec/ui/csh/wfirot.tk
/kroot/kss/nirspec/ui/csh/wfslit
/kroot/kss/nirspec/ui/csh/wfslit.tk
/kroot/kss/nirspec/ui/csh/wftest
/kroot/kss/nirspec/ui/csh/wftest.tk
/kroot/kss/nirspec/ui/csh/wftest2
/kroot/kss/nirspec/ui/csh/wftest2.tk
```

An example of how to use these is given below:

```
alias m "modify -s nirspec"
# Move the slit, echelle and cross disp. simultaneously.
m slitpos=0
m echlpos=62.5
m disppos=35.2

# Wait for the three mechansism to move.
wfslit
wfechl
wfdisp

# Begin a 1024 exposure
m go=1

# Wait for the exposure to finish
wfg

# Put a comment into the dataviews status window.
m comment='Script is finished'
```

This script does just what the comments say. First it creates a handy alias for modify. Then it gives move commands to three mechanisms. They will move together and complete at different times. Then we issue the wf commands in sequence. The **wfslit** script will halt execution until the slit has finished moving. Then the **wfechl** command will be executed to wait for the echelle. If it has already completed its motion, then **wfechl** will return immediately and **wfdisp** will be executed. When all three mechanisms have completed, a *go* command is issued to start a 1024 exposure. Then a **wfg** script is started to wait for the exposure to end. Finally, a comment is set which will appear in the next frames header unless it is replaced, but which also appears in the dataviews window.

5. Miscellaneous Scripts

There are several scripts that have been written for special purposes. These can be called by other scripts, but more typically they are stand alone. Here we give brief discriptions of each:

`/kroot/kss/nirspec/ui/csh/alerttemp.tk`

This tk popup tells the user that the cabinet temperature is above 40 degrees C, and therefore the power to all but the root transputer are shutdown.

`/kroot/kss/nirspec/ui/csh/warntemp.tk`

This tk popup warns the user that the cabinet temperatures are above 35 C, so there is a coolant problem.

`/kroot/kss/nirspec/ui/csh/box9`

This csh script takes 9 scam exposures around a regular grid of locations on the sky, then it spawns an IDL process that reduces these images into a single 256x256 image that can be used to move faint objects onto the slit.

`/kroot/kss/nirspec/ui/csh/cabtemps`

This csh script compiles all of the cabinet temperatures and displays them.

`/kroot/kss/nirspec/ui/csh/centslit`

This csh script takes an x, y position on the scam and moves whatever is at that location to the center of the current slit.

`/kroot/kss/nirspec/ui/csh/check_lamps`

This csh scripts cycles through each lamp individually and then reads the sensors. It displays the sensor readings for each lamp so broken lamps can be identified.

`/kroot/kss/nirspec/ui/csh/diskfull.tk`

A tk popup that the server uses to notify the user when the data disk is full.

`/kroot/kss/nirspec/ui/csh/emerg_power_off`

This csh script cycles off all but the root transputer power. It is run by the server if the cabinet temperatures rise above 40 C.

`/kroot/kss/nirspec/ui/csh/emerg_power_on`

This csh script can turn on all power after an emerg_power_off. It requires the root transputer power to be on, and the transputer connection still be alive. After a power up, the server must be restarted to reload the transputer code.

`/kroot/kss/nirspec/ui/csh/init_filters`

Since both filter wheels cannot be moved at the same time, we created a special scipt to initialize the filter wheels one after the other. It is used by the dataviews when you select to init the filter wheels.

`/kroot/kss/nirspec/ui/csh/malign`

This routine takes a series of scam images with the keck mirror segments turned out. It allows for the mirrors to be stacked and focused.

`/kroot/kss/nirspec/ui/csh/malign.tk`

This tk popup is used by the malign script to notify the user about the progress of the malign script.

`/kroot/kss/nirspec/ui/csh/mxy`

This csh script allows the user to move objects around the scam field in scam pixel coordinates.

`/kroot/kss/nirspec/ui/csh/reboot`

This csh script first queries the server for all of the motor positions, then stops the software and restarts the server. For all of the mechanisms that were “ok”, they are initialized and moved to their previous locations. For the mechanisms that were having problems, they are not initialized, but the user is told about which mechanisms might cause troubles.

`/kroot/kss/nirspec/ui/csh/slitmove`

Moves the object in slit coordinates.

`/kroot/kss/nirspec/ui/csh/snapi`

Takes an exposure at the current location, then moves by node and nodn coordinates and takes a second exposure. It then spawns an IDL routine that subtracts these two frames and displays the difference.

`/kroot/kss/nirspec/ui/csh/update_slit`

This csh script reads the slit x,y and angle from the server and sets the appropriate coordinate systems in the dcs so a pointing origin is set up for the slit.

6. EFS Scripts

The echelle format simulator (EFS) is not directly connected to NIRSPEC server, so to control the instrument it creates scripts that are either saved or immediately executed. If you save the configuration by selecting "Save Configuration As..." under the "FILE" menu, then you may use any directory and filename for saving. These files can be read back using the "Open Configuration ..." option also under the "FILE" menu. EFS scripts should never be modified by hand, because the EFS doesn't actually read the csh code, but instead uses a very cryptic header to store and recover its parameters. If you edit an EFS script, you'll see a series of lines at the top of the file like:

```
#!/bin/csh -f
# 1
# 3
# 2
#  1.10180378
#  0.17186975
# 1
#high res
#                0
...
```

This a direct output of the EFS internal variables, and without digging into the EFS code, there is no way of really knowing which number is which. But it does allow for easy reading and writing of scripts by the EFS without developing a csh parser. So you can modify the csh code of an EFS script by hand, but don't expect the EFS to read it in correctly. In most cases you should just read the script back in and use the EFS to modify and save it.

EFS scripts can be run at the command line by sourcing them as described above, but the easiest way is to load it into the EFS and then press the **go** button. A new script is generated as part of a numbered sequence that provides a reconstructable history of what you did during your observations.