NAME

nclsh − Wrapper for NCL interpreter providing simplified passing of parameters

SYNOPSIS

nclsh [*FILE*] [*OPTION*]... [*ARG*]...

DESCRIPTION

This is a replacement for calling the NCAR Command Language (NCL) directly whenever you need to pass command line arguments to NCL without twisting your fingers into a knot.

It simply translates the parameters into a form recognized by NCL, and then runs the **ncl** command as configured on your system. Within nclsh, ncl is run with the '−n' option, i.e. print output is not prefixed by element indices.

Parameter translation

Any command line parameter is parsed and put into NCL syntax according to these rules:

- Parameters beginning with '+', '−' or '−−' are options.
- Options that are continued with '=' take the remainder of the parameter as value
- Values may be scalar or arrays; array elements are separated by ',' or ':'.
- Values are of type boolean ('True' or 'False', case-insensitive), numeric, or string.
- Options without value are taken to be boolean. Prefix '+' assigns 'False', prefix '−' or '−−' assigns 'True'.
- If an option without value begins with '−no' or '−−no', and the option name is longer than 2 characters, the leading 'no' is stripped from the option, and its value becomes 'False'.
- The first non-option parameter is taken to be the NCL script file name. Its treatment is described below
- Any remaining non-option parameters are assigned to a string array option named 'ARGS'. Therefore, 'ARGS' is not a valid option name for user options. If there are no such parameters, 'ARGS' is undefined.
- All options are passed to NCL as variable assignment. Variable name is the option name without the '+', '−', '−−', or 'no' prefixes. Scalar values are assigned as NCL scalar, array values with NCL array syntax. All non-numeric and non-boolean values are surrounded by NCL's string quote '"'.
- For array assignments, all elements must have the same type. So, if elements are not recognized to be all numeric or all boolean, all elements in that array are interpreted as strings and quoted.

Also note that, when calling nclsh, the usual shell restrictions apply when you want to embed white space or meta-characters into your string values.

Examples

```
nclsh −format=pdf            ->  ncl format="pdf"
nclsh −−format=pdf           ->  ncl format="pdf"
nclsh +format=pdf            ->  ncl format="pdf"
nclsh −values=12,34.56,78e9  ->  ncl values=(/12,34.56,78e9/)
nclsh −variables=slp:sst     ->  ncl variables=(/"slp","sst"/)
nclsh −verbose               ->  ncl verbose=True
nclsh +verbose               ->  ncl verbose=False
nclsh −noverbose             ->  ncl verbose=False
```

Please note:

```
nclsh −no                    ->  ncl no=True
nclsh +no                    ->  ncl no=False
nclsh +noverbose             ->  ncl noverbose=False
nclsh −values=12,True,78e9   ->  ncl values=(/"12","True","78e9"/)
```

Script file

If a NCL script name has been recognized in the parameter list, nclsh checks if the first line of this file contains the she-bang (#!) sequence. If this is the case, the whole script is copied to a temporary file with the first line commented out by NCLs comment character ';'. This allows using nclsh as command script interpreter. The recommended way to do this is heading your NCL script with the following line:

```
#! /usr/bin/env nclsh
```

Note that, in this case, any error message refers to the temp file name instead of the original script file name. However this should make no difference when using line numbers as only the first line is altered and no lines are deleted or added.

SEE ALSO

The NCL website <http://www.ncl.ucar.edu/>, providing all you ever wanted to know about NCL, and possibly more.

AUTHOR

Written by Karl-Hermann Wieners, but ultimately inspired by Ralf Müller.

COPYRIGHT