Introduction to NCL

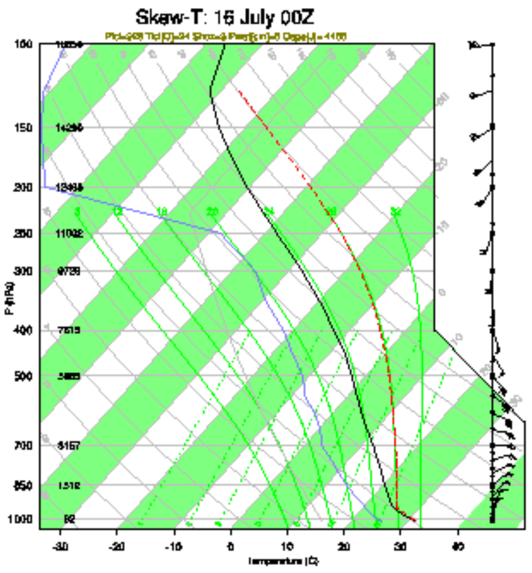
[part 1 of 3]

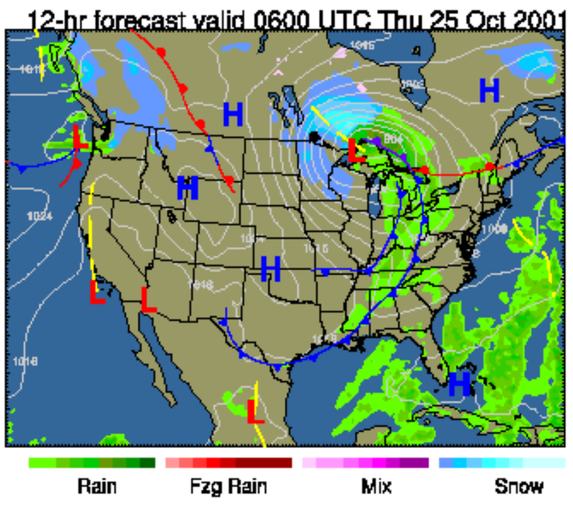
Dennis Shea

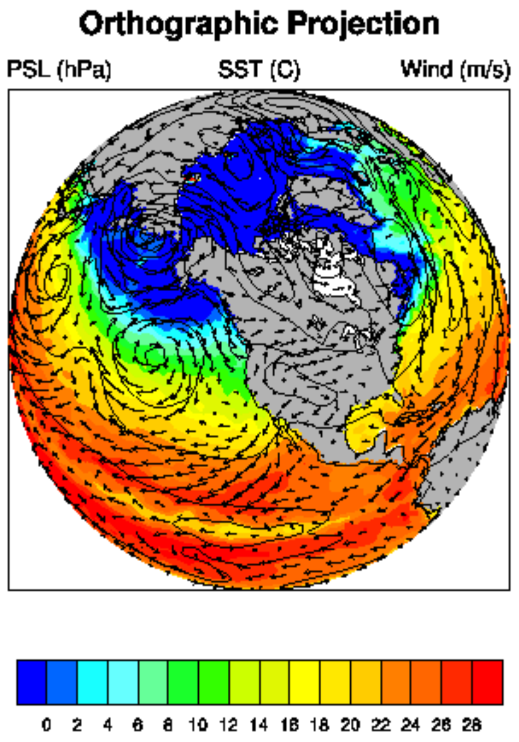
NCL
NCAR Command Language

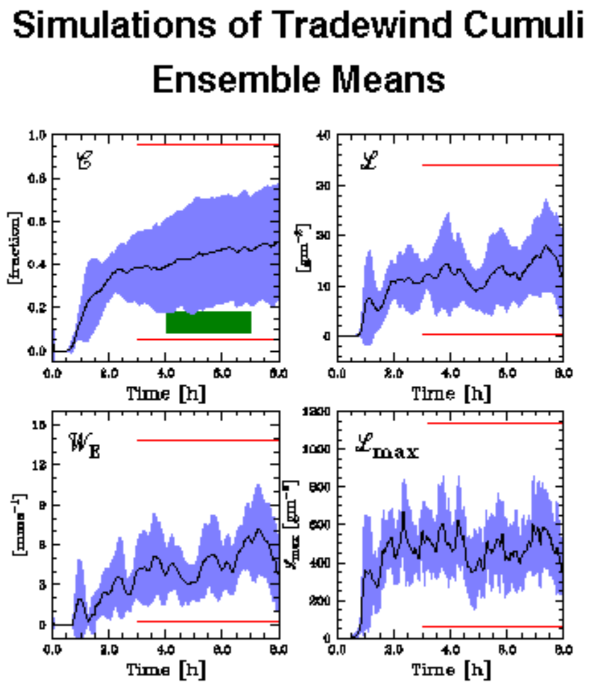
NCAR

Sponsored by the National Science Foundation









Introduction: Key Points

NCL gross pictoral overview

NCL script creation and execution

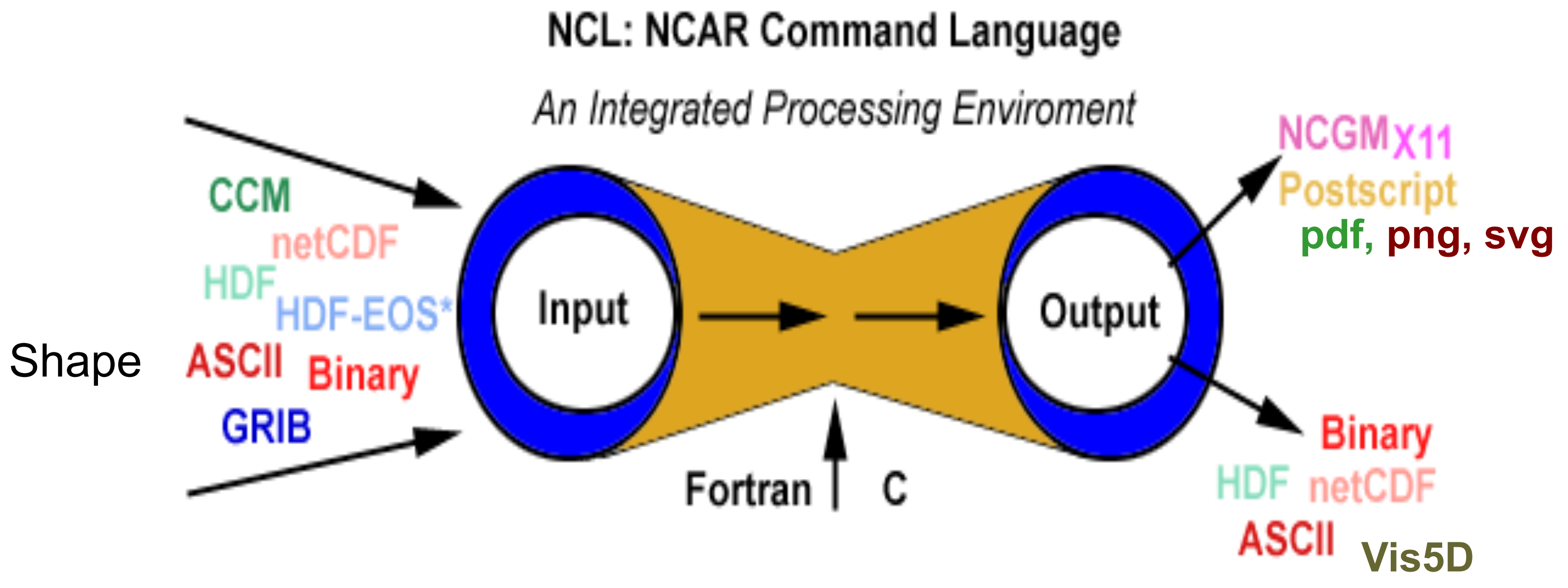
NCL syntax characters

= , :=, ->, @, !, &, $, ; [… additional syntax]

NCL variable types: create and delete

NCL Pictoral Overview

- **Integrated data processing environment**



- **freeware: supported, public domain**
- **portable: *nix, windows (cygwin), MacOS**
- **general purpose: unique capabilities, functions**
- **excellent 2D graphics** (limited 3D)

Executing NCL: Interactive (1 of 3)

- **Interactive Mode** **(unix/linux command line)**
  - **ncl** *[options][command-line-arguments]* <return>

    ncl> enter commands

    ncl> **quit** <return>
  - can save (record) interactive commands

    ncl> **record** “file_name”

    ncl> …. enter commands …

    ncl> **stop record**

- **Interactive environment**
  - use for simple testing
  - can use ‘up/down arrow’ to recall previous lines
  - **not** as ‘friendly’ as (say) IDL, Matlab, ferret
    - not good at error recovery

Running NCL: Batch (2 of 3)

Recommended

- **Batch Mode** [**.ncl** suffix is **optional**]
  - **ncl** *[options][arguments]* **script**.ncl
    - **ncl <** script.ncl [also acceptable]
  - **ncl** *[options][arguments]* **script**.ncl **>&!** out
  - **ncl** *[options][arguments]* **script**.ncl **>&!** out **&**
    - appending "**&**" means put in background
    - note: the **>&!** **&** are **csh** and **tcsh** syntax

- **NCL built for larger processing tasks**
  - better accomplished via a **script** (recommended)
    - use editor (**vi**, **nedit**, **emacs**, …)
      - editor language enhancements **(**Under 'Support'**)**
    - enter/delete statements; save file
    - run the script as above

Running NCL: predefined options (3 of 3)

- **ncl –hnxV script.ncl**
  - [predfined options are preceded by dash]
- may be used for interactive or batch mode
- **Information**
  - **ncl –h** [display predefined options and usage and exit]
  - **ncl –V** [print the NCL version and exit]
- **Action**
  - **ncl –x** [echo statements as encountered (debug)]
  - **ncl –n** [don't enumerate dimensions of values in **print**()]
- Multiple options may be specified
  - **ncl –nx** [not ncl –n –x]

- **Experiment** with options (for fun)

NCL Syntax Characters (subset)

- **=** - assignment
- **:=** - **re**assignment (v6.1.2)
- **;** - comment [can appear anywhere; text to right **;** ignored]
- **->** - use to (im/ex)port variables via **addfile(s)** function(s)
- **@** - access/create attributes
- **!** - access/create named dimension
- **&** - access/create coordinate variable
- **{…}** - coordinate subscripting
- **$...$** - enclose strings when (im/ex)port variables via addfile(s)
- **(/../)** - array construction (variable); remove meta data
- **[/../]** - list construction;
- **[:]** - all elements of a list
- **:** - array syntax
- **|** - separator for named dimensions
- \ - continue character [statement to span multiple lines]
- **::** - syntax for external shared objects (eg, fortran/C)

Data Types

numeric (classic netCDF3)

double	(64 bit)
float	(32 bit)
long	(64 bit; signed +/-)
integer	(32 bit; signed +/-)
short	(16 bit; signed +/-)
byte	(8 bit, signed +/-)

- complex **NOT** supported

non-numeric

- string
- character
- graphic
- file
- logical
- list

enumeric (netCDF4; HDF5)

int64	(64 bit; signed +/-)
uint64	(64 bit; unsigned)
uint	(32 bit; unsigned)
ulong	(32 bit; unsigned)
ushort	(16 bit; unsigned)
ubyte	(8 bit, unsigned)

snumeric

[numeric , enumeric]

Simple Variable Creation

```
a_int      = 1
a_float    = 2.0                    ; 0.00002 , 2e-5
a_double = 3.2d                     ; 0.0032d , 3.2d-3
a_string   = "a"
a_logical = True [False]            ; note capital T/F
```

- **array constructor characters (/…/)**
  - a_integer = (/1, 2, 3/) ; **ispan**(1,3,1)
  - a_float = (/2.0, 5 , 8.0/) ; **fspan**(2,8,3)
  - a_double = (/12 , 2d0 , 3.2 /) ; (/12,2 ,3.2 /)*1d0
  - a_string = (/"abcd", "e", "Hello, World"/)
  - a_logical = (/True, False, True/)
  - a_2darray = (/ (/1,2,3/), (/4,5,6/), (/7,8,9/) /)

Variable Creation and Deletion

```
    a   = 2.0
    pi  = 4.0*atan(1.0)
    s   = (/ “Melbourne”, “Sydney”, “Toulouse”, “Boulder” /)
    r   = f->precip                              ; (time,lat,lon)
    R   = random_normal(20,7, (/N,M/) )          ; R(N,M)
    q   = new ( (/ntim, klev, nlat, mlon/), “double” )
; free memory; generally, do not need to do this
; delete each variable individually
    delete(a)
    delete(pi)
    delete(s)
    delete(r)
    delete(R)
; delete multiple variables in one line
    delete(  [/ a, pi, s, r, R, q /] )           ; [/…/] list syntax
```

Conversion between data types

- **NCL** is a 'strongly typed' language
  - constraints on mixing data types
- **coercion**
  - implicit conversion of one type to another
- **automatic coercion when no info is lost**
  - let i be integer and x be float or double
  - fortran: x=i   and   i=x
  - NCL: x=i   and   i=**toint**(x)
- **many functions to perform conversions**

Variable Reassignment

- **NCL = will not allow the following**

```
k  = (/ 1, 3, 4, 9 /)      ; 1d array, type integer
      … later in code …
k  = (/17.5, 21.4/)        ; different size and type
```

- **Two approaches**
  - Up to version 6.1.1, **2 steps required**
    - delete(k)   ; delete existing variable
    - k = (/17.5, 21.4/)   ; new assignment
  - version 6.1.2
    - k := (/17.5, 21.4/)   ; delete previous variable ; and reassign 'k'

- **NCL := will not allow the following**

```
x  := x(::4,:,:)            ; same variable
```