Basics I

Scripting Language

Contents

- Interactive mode
- Special syntax characters
- Expressions: Algebraic and logical operators
- Data types
- Variables
- Attributes
- Named dimensions and coordinate variables
- Statements and loops
- print and printVarSummary
- Create and run a NCL script

Interactive mode

Start NCL

```
>  ncl
 Copyright (C) 1995-2015 - All Rights Reserved
 University Corporation for Atmospheric Research
 NCAR Command Language Version 6.3.0
 The use of this software is governed by a License Agreement.
 See http://www.ncl.ucar.edu/ for more details.
ncl 0>
```

NCL prompt

`ncl -Q` turns off echo of NCL version and copyright infos

Exit NCL

```
ncl 1>  quit
or
ncl 1>  exit
```

Special syntax characters

=	assignment syntax
:=	reassignment operator
;	begins a comment
->	use to im- or export variables via addfile function
@	create or reference attributes
!	create or reference named dimension
&	create or reference coordinate variable
$...$	enclose strings when im- or export variables via addfile
(/…/)	array constructor
[/…/]	list constructor
{…}	coordinate subscripting
:	array syntax, e.g. b(:) means all elements of array b
|	seperator for named dimensions
\	continue character
::	syntax for external shared objects (e.g. fortran/C)

Algebraic operators

+	Addition
-	Substarction / Negation
*	Multiplication
/	Division
%	Modulus
>	Greater than
<	Less than
^	Exponentiation
#	Matrix multiplication

Logical operators

.lt.	Less than
.le.	Less equal
.eq.	Equal
.ne.	Not equal
.ge.	Greater equal
.gt.	Greater than
.and.	And
.or.	Or
.xor.	Exclusive or
.not.	Not

Data types

Numeric data types

double	(64 bit)
float	(32 bit)
long	(32 bit or 64 bit; signed +/-)
integer	(32 bit; signed +/-)
short	(16 bit; signed +/-)
byte	(8 bit; signed +/-)
complex	NOT supported

Enumeric data types

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

Non-numeric data types

string
character
graphic
file
logical
List

Snumeric

numeric, enumeric

Variable creation

```
x     = 0.12                      ; float
y     = -4                        ; integer
z     = 20.d                      ; double
title = "This is the title string" ; string
a     = True                      ; logical
```

Array constructor characters (/../)

```
a     = (/1, 2, 3, 4/)            ; integer array
b     = (/1, 2.0, 3.0, 4./)       ; float array
c     = (/1., 2, 3., 4.0/) * 1d5  ; double array
d     = (/"hello", "world"/)      ; string array
e     = (/True, False, False, True/) ; logical array
f     = (/ (/1,2/), (/3,4/), (/5,6/) /) ; 2D array
```

List constructor characters [/../]

```
l     = [/x,a,title/]             ; list
```

Delete a variable

```
delete(title)          delete( [/ a,b,c,d,e,f /] )
```


Variable subscripting (1)

Array index starts with 0

Array *a* containing 6 elements: `a = (/2,7,3,6,0,9/)`

Standard subscript m:n:I range m to n in strides of i

```
Reverse the array:        aR = a(::-1)
                                aR = (/9,0,6,3,7,2/)
Select 1st element:       x = a(0)
                                x = 2
Select 2nd–5th elements:  x14 = a(1:4)
                                x14 = (/7,3,6,0/)
Select first 4 elements:  x03 = a(:3)
                                x03 = (/2,7,3,6/)
Select every second element:
                          x2 = a(::2)
                                x2 = (/2,3,0/)
Select every second value within element 2 and 5:
                          x3 = a(1:4:2)
                                x3 = (/7,6/)
```

NCL

Variable Subscripting (2)

- Row major
  - left dimension varies slowest
  - right varies fastest
  - dimension numbering from left to right

Standard subscript m:n:I range m to n in strides of i

3D Array *slp(time,lat,lon)*

```
slp                 → the entire array
slp(0,:,:)          → 1st time step all lat, all lon
slp(:,:,::5)        → all time steps, all lat values,
                      every 5th lon value
slp(:3,::-1,15:30)  → first 4 time steps, reverse lat
                      index 15-30 of lon
```

Coordinate Variable Subscripting

```
slp(0,{-45},{-60:60:5})  → 1st timestep, nearest to
                           lat -45°, nearest to lon
                           -60° to 60° in steps of 5
```


NCL

Variable Subscripting (3)

- array index starts with 0
- (rows x columns)

2D Array *f = (/(/1,8/),(/3,0/),(/8,2/)/)*

1	8
3	0
8	2

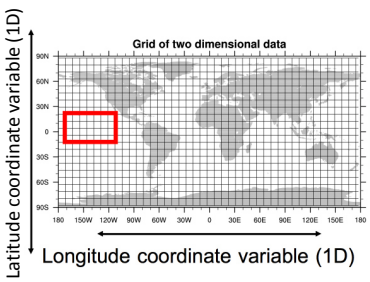
```
f                   → entire array

f(0,:)              → 1st row, all columns => 1 and 8
f(1,:)              → 2nd row, all columns => 3 and 0
f(2,:)              → 3rd row, all columns => 8 and 2

f(:,0)              → all row, column 1  => 1, 3 and 8
f(:,1)              → all row, column 2  => 8, 0 and 2
```


NCL Language

Variable subscripting (4)



Standard:
T(9:13,1:8)

Coordinate:
T({-10:20},{-170:-110})

Combined:
T({-10:20}, 1:8)

Graphic: NCL workshop from NCAR

Conversion between data types

- **Strongly typed language** **Temp** is NOT equal to **temp**
- **Coercion**
  - **Implicit conversion of one type to another**
- **Automatic coercion when no information is lost**

```
i = 2
x = 4.5
x = i        →  no information is lost (x: float)
i = x        →  error
fatal:["NclVar.c":1390]:Assignment type mismatch, right hand side
can't be coerced to type of left hand side
fatal:["Execute.c":8565]:Execute: Error occurred at or near line 2

i = toint(x)  →  i = 4
```

- **Conversion functions**
  - **toint, tofloat, todouble, tolong, toshort, tostring, tobyte,……**

Attributes

- **Info about a variable or file (called meta data)**
- **Attributes can be of any data type except file or list**
- **Scalar, multi dimensional array (string, numeric)**

To assign an attribute

```
lon@units          = "degrees_east"
t@long_name        = "Near-Surface Air Temperature"
time@units         = "days since 1949-12-01 00:00:00"
temp@_FillValue    = 1e20
temp@missing_value = 1e20
```

Many attribute functions available, e.g.

```
isatt
getfilevaratts
```

Delete an attribute

```
delete(time@units)
```


Attribute _FillValue

- **Reserved attribute (Unidata and NCL)**
- **netCDF-CF convention compliant**
- **Most NCL functions recognize the _FillValue**

If *missing_value* is set, the attribute *_FillValue* must be the same value. If *missing_value* is set and *_FillValue* is undefined, NCL will create it internally

```
slp@_FillValue = slp@missing_value
```

Function *ismissing* checks for _FillValue in the data

```
if(any(ismissing(slp)))then
   …do anything…
end if
```

! Recommended: Do not use zero as a _FillValue !

Arrays

NCL/C/C++	0-based and row major
Fortran	1-based and column major

- Row major
  - left dimension varies slowest
  - right varies fastest
  - dimension numbering from left to right
- Subscripts are zero based
  N values: subscripts from index 0 to index N-1
  e.g. array x has 12 elements the
  first element x(0)
  last element x(11)

```
T(12,5,4)    left   dimension index 0 with 12 elements
                    (varying slowest)
             middle dimension index 1 with 5 elements
             right  dimension index 2 with 4 elements
                    (varying fastest)
```

Dimension of T (3D)

Named dimensions

Shape: number of dimensionen
Dimension Size: number of elements of each dimension

A 1D variable with one value is called scalar variable.

Dimensions are read from left to right, 0 to N-1 like arrays.
slp(:,:,:) → slp(0,1,2) → 3 dimensionale Variable

Assigning 'named dimensions' to a 3D Variable slp with the ! sign:

```
slp!0 = "time"
slp!1 = "lat"
slp!2 = "lon"
```

! point to the name of the dimension

Reorder (reshape) of an array
tas(lon,lat,time) → tas(time|:, lat|:, lon|:)

Coordinate variables

- 1D variable with the same name as a dimension, which names the coordinate values of the dimension
- must not have any missing data
- must be strictly monotonic (values increasing or decreasing)

E.g. 2D array temp(4,5)

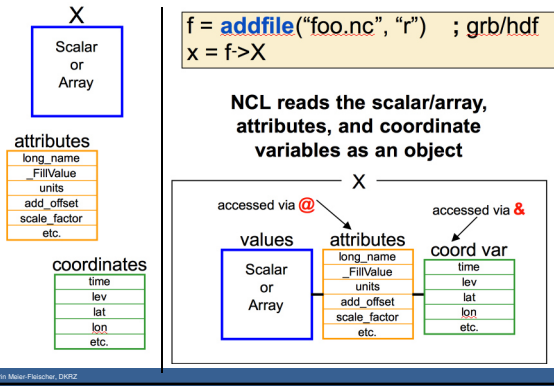
```
temp!0 = "lat"              ; left dimension
temp!1 = "lon"              ; right dimension

lon_pts = (/ 0., 15., 30., 45., 60. /)  ; size 5
lat_pts = (/ 30., 40., 50., 60. /)      ; size 4
lon_pts@units = "degrees_east"
lat_pts@units = "degrees_north"

temp&lon = lon_pts
temp&lat = lat_pts
```

@ points to an attribute of a dimension or variable
& commit the data to the dimension or variable

NCL: netCDF variable model



Graphic c. NCL workshop/from NCAR

Statements and loops (1)

Statements

- **Blocks (a group of statements; begin-end)**

```
begin
        statement 1
        statement 2
        ……
end
```

- **Conditional expressions (if-then, if-then-else)**

```
if ( scalar_logical_expression ) then
        [statement(s)]
else
        [statement(s)]
end if
```

No direct else-if
→ else if construction

```
if (…) then
   …
else if(…) then
   …
end if
end if
```


Statements and loops (2)

Statements

- **Loops (do, do-while)**

```
do n=start,end[,stride]
        [statement(s)]
end do
```

  - **Loop while a logical expression is True:**

```
do while (scalar_logical_expression)
        [statement(s)]
end do
```

- **Assignments / Reassignments**

```
var = "This is a string"          ;-- var of type string
var := (/1.0,10.0,15.0/)          ;-- var of type float
```

Statements and loops (3)

Statements

- **Procedures**

```
undef ("procedure_name")
procedure procedure_name(declaration_list)
local local_identifier_list
begin
   statement_list
end
```

- **Functions**

```
undef ("function_name")
function function_name(declaration_list)
local local_identifier_list
begin
   statement_list
   return(return_value)
end
```


printVarSummary, print, write_matrix (1)

- **printVarSummary** - information about variable

printVarSummary(tsurf)

```
Variable: tsurf
Type: float
Total Size: 2949120 bytes
            737280 values
Number of Dimensions: 3
Dimensions and sizes:  [time | 40] x [lat | 96] x [lon | 192]
Coordinates:
            time: [   0.. 234]
            lat: [88.57216851400727..-88.57216851400727]
            lon: [-180..178.125]
Number Of Attributes: 5
  long_name :  surface temperature
  units :      K
  code :       169
  table :      128
  grid_type :  gaussian
```


printVarSummary, print, write_matrix (2)

- **print** - same info as printVarSummary
  - print values

ndims = dimsizes(tsurf)

print(ndims)

Variable: ndims
Type: integer
Total Size: 12 bytes
3 values
Number of Dimensions: 1
Dimensions and sizes: [3]
Coordinates:
(0) 40
(1) 96
(2) 192

print("ndims = "+ndims)

(0) ndims = 40
(1) ndims = 96
(2) ndims = 192

printVarSummary, print, write_matrix (3)

Embedded strings

```
print("Min temp = " + min(temp) + "  Max temp = " + max(temp))
```

→ `Min temp = 21.7  Max temp = 37.1`

Formated printing

```
print("Value  x = " + sprintf("%5.2f",x))
```

→ `Value x =  6.87`

Leading zeros

```
fn = "file_" + sprinti("%0.5i",2) + ".nc"
print("file name = "+fn)
```

→ `file name = file_00002.nc`

printVarSummary, print, write_matrix (4)

Other print functions

`write_matrix(variable,format,opt)`	pretty-print 2D array to stdout
`print_table(list)`	formatted print of all elements from a list
`printMinMax(variable,0)`	prints the minimum and maximum value of a variable
`printFileVarSummary(file,variable)`	prints a summary of a file variable's information

Create and run a NCL script (batch mode)

Create a NCL script, e.g. *myscript.ncl*

Since NCL version 6.2.0 not needed

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin
   print("Hello World")
   x = 1.5
   print(x)
end
```

Run the script in a terminal window

```
ncl myscript.ncl
```

See what happens

NCL

Math Functions (1)

- abs
- avg
- cos, cosh
- sin, sinh
- tan, tanh
- acos, asin
- atan, atan2
- exp
- log, log10
- mod
- round
- sqrt
- sum
- ……..

NCL

Math Functions (2)

```
ival =  -12
k    =  abs(ival)
print(k)                    → 12

f      =  12.5
sqrt_f =  sqrt(f)
print(sqrt_f)               → 3.535534

s    =  0.5
sval =  sin(s)
print(sval)                 → 0.4794255

a       =  (/(/1,2,3/), (/4,5,6/), (/7,8,9/)/)
average =  avg(a)
print(average)          → 5
```


NCL

NCL Analysis Functions

(more than 600 built-in analysis and visualization functions)

Earth Science:	
Climatology	seasonal means, standard deviations of monthly means, daily/monthly anomalies of daily data climatology, long term daily means, daily from monthly climatology, …
CESM	Functions for Community Earth System Model
Date	Date conversion and formatting routines, …
Lat/Lon functions	Generates Gaussian latitudes, land sea mask, reorder longitude array, …
Metadata/missing values	Copy/delete metadata, set/get missing value, set/get attributes, …
Meteorology	Zonal mean, weighted average, potential vorticity, sea level pressure, …
Oceanography	Convert ocean depth to pressure, remap POP grid, …
WRF functions	Specific functions and procedures for WRF ARW model data

NCL Analysis Functions

Arithmetic and statistics:	
Arithmetic functions	sin, cos, tan, atan, atan2, averages, variance, min/max, …
Cumulative distribution functions	Binominal density, number of success, number of binominal trials, …
Empirical orthogonal functions	Calculates empirical orthogonal functions via a correlation matrix, …
ESMF regridding	From rectilinear, curvilinear, and unstructured grid to any of these types
Interpolation	Bilinear, cubic spline, natural neighbor, inverse distance weighted, …
Random number generators	Pseudo random numbers and 2D arrays, using gamma distribution, …
Regridding	rectilinear, curvilinear, unstructured, area conserve, local area, …
Singular value decomposition	Singular value decomposition to return the left and right homogeneous and heterogeneous arrays associated with the two input datasets, …
Spherical harmonics	Wind components via spherical harmonics, given vorticity and divergence on a fixed grid, …