# Introduction to Kernels
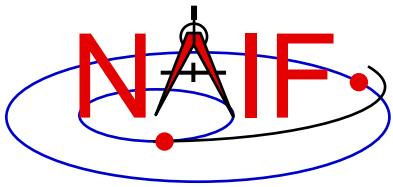
## June 2019

## (Class version)

**Navigation and Ancillary Information Facility**

# "Kernel" means file

## "Kernel" means a file containing ancillary data

**"Kernel" means a file containing "low level" ancillary data that may be used, along with other data and SPICE Toolkit software, to determine higher level observation geometry parameters of use to scientists and engineers in planning and carrying out space missions, and analyzing data returned from missions.**
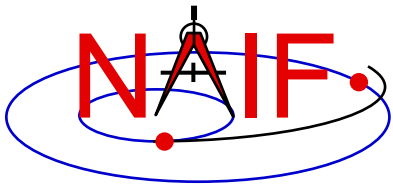
# SPICE Kernels Family

- **S**PK
  - **Spacecraft and Planet Ephemeris**
- **P**CK
  - **Planetary Constants, for natural bodies**
    - » **Orientation**
    - » **Size and shape**
- **I**K
  - **Instrument**
- **C**K
  - **Orientation ("Camera-matrix")**
- **E**K
  - **Events, up to three distinct components**
    - » **ESP: science plan**
    - » **ESQ: sequence**
    - » **ENB: experimenter's notebook**

EK is rarely used

- **FK**
  - **Reference frame specifications**
- **SCLK**
  - **Spacecraft clock correlation data**
- **LSK**
  - **Leapseconds**

- **MK**
  - **Meta-Kernel (a.k.a. "FURNSH kernel")**
  - **Mechanism for aggregating and easily loading a collection of kernel files**

- **DSK**
  - **Digital shape kernel**
    - » **Tesselated plate model**
    - » **Digital elevation model (under development)**

- **DBK**
  - **Database mechanism**
    - » **Primarily used to support the ESQ**
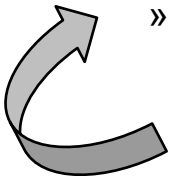
# SPICE Kernel Forms

- **Binary form**
  - **A file mostly containing data encoded in binary form**
  - **Provides rapid access to large amounts of numeric data**
  - **Binary kernels are not human-readable; they require the use of Toolkit software to examine the data contents**

- **Text form**
  - **Plain text files containing only printing characters (ASCII values 32-126), i.e. human-readable text.**
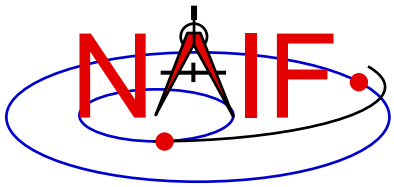
# Text and Binary Kernels
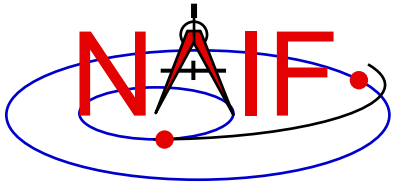
**SPICE text kernels are:**

- text PCK (the most common type of PCK)
- IK
- FK
- LSK
- SCLK
- MK

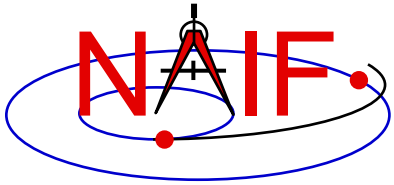**SPICE binary kernels are:**

- SPK
- binary PCK (has been used only for Earth, Moon and Eros)
- CK
- DSK

- ESQ (part of the E-kernel)
- DBK (database kernel)

Rarely used

# Kernel Architecture

- Text kernels
- Binary kernels
- Comments in kernels

# Text Kernel Contents

- **A text kernel is a plain text file of ASCII data**

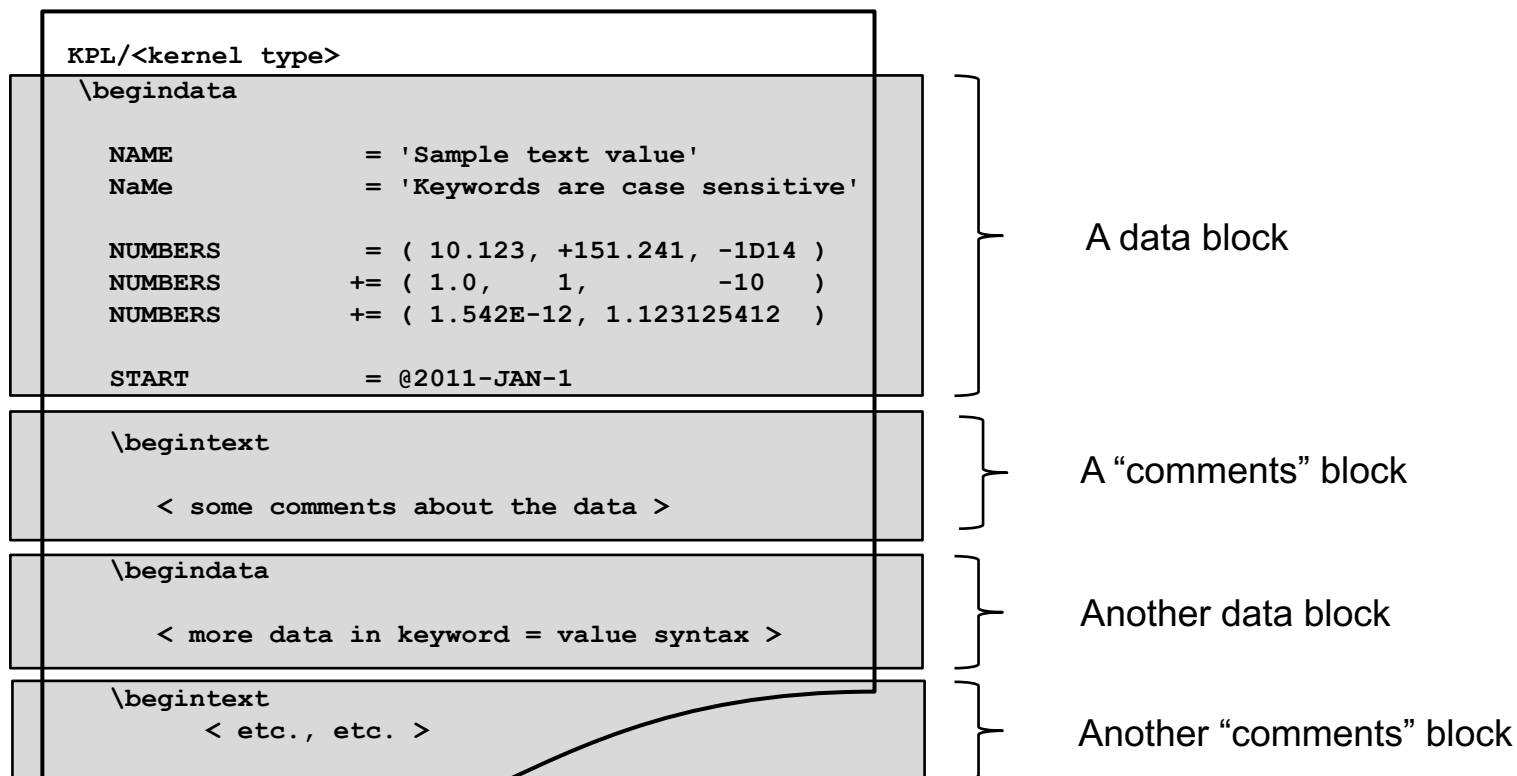- **It contains assignments of the form:**

  ```
  variable_name = value(s)
  ```

- **A text kernel should also contain descriptive comments that describe the assignments**
  - **Comments are sometimes referred to as "meta-data"**
    - » **Don't confuse this usage with the "meta-kernel" described later in this tutorial**

# Example Text Kernel

```
KPL/<kernel type>

 \begindata

  NAME            = 'Sample text value'
  NaMe            = 'Keywords are case sensitive'

  NUMBERS         = ( 10.123, +151.241, -1D14 )
  NUMBERS        += ( 1.0,     1,          -10   )
  NUMBERS        += ( 1.542E-12, 1.123125412  )

  START           = @2011-JAN-1
```
A data block

```
 \begintext

    < some comments about the data >
```
A "comments" block

```
 \begindata

    < more data in keyword = value syntax >
```
Another data block

```
 \begintext
       < etc., etc. >
```
Another "comments" block
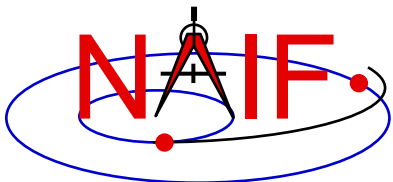
- **The next several pages describe what you see above**
- **See the "Kernel Required Reading" document for details**
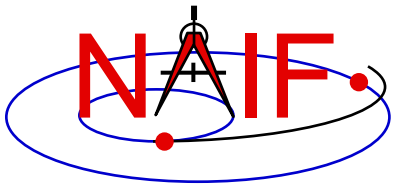
# Text Kernel Formatting

- **`KPL/<text kernel type>`**
  - Its use is optional, but is highly recommended
  - Must appear on the first line, starting in column 1
  - Tells SPICE software what kind of kernel it is
  - Text kernel types are FK, IK, PCK, SCLK, MK

- **`\begindata`** and **`\begintext`**
  - Markers, on lines by themselves, which set off the beginning of data and the beginning of comment (metadata) blocks respectively
  - They need not begin in column 1

- **`<LF>`** for Unix/Linux/Mac or **`<CR><LF>`** for Windows
  - End of line marker (usually not visible when displaying a text kernel)
  - Must be present on EVERY line in the text kernel

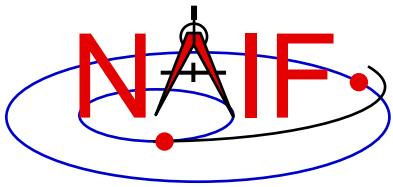- **Max line length, including any white space, is 132 characters**
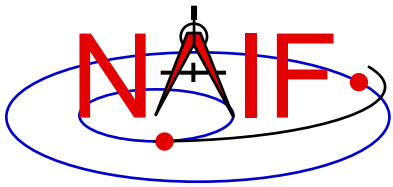
# Text Kernel Operators

- **An assignment using the "=" operator associates one or more values with a variable name.**

- **An assignment using the "+=" operator associates additional values with an existing variable name.**

- **An assignment using the "@" symbol associates a calendar date with a variable name.**
  - The string will be parsed and converted to an internal double precision representation of that epoch as seconds past the J2000 epoch
    - » There is no time system implied
    - » This conversion does not need a leap seconds kernel
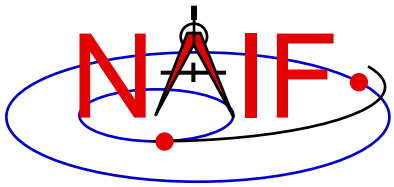
# Variable Names and Values

- **Variable names**
  - **Max of 32 characters**
  - **Are case sensitive (recommendation: use only upper case)**
  - **Cannot include space, period, parenthesis, equals sign or tab**
  - **Recommendation: don't use the "+" sign as the last character**

- **Values**
  - **Numeric:  integer, fixed point and scientific notation are allowed**
  - **String:**
    - » **enclosed in single quotes**
    - » **maximum length of 80 characters on a given line**
      - **SPICE has means to concatenate multiple string values to allow for values exceeding 80 characters**
    - » **string values may contain any printing ASCII character, including blank**
  - **Time:  identified by the "@" character**
  - **Any of these three types can be provided as an n-dimensional vector of values**
    - » **Components are separated by commas or white space (but not TABs)**
    - » **Parentheses are used to enclose the vector**
    - » **Each string value in a vector is contained in single quotes**
    - » **Values in a vector must all be of the same type (numeric, string or time)**

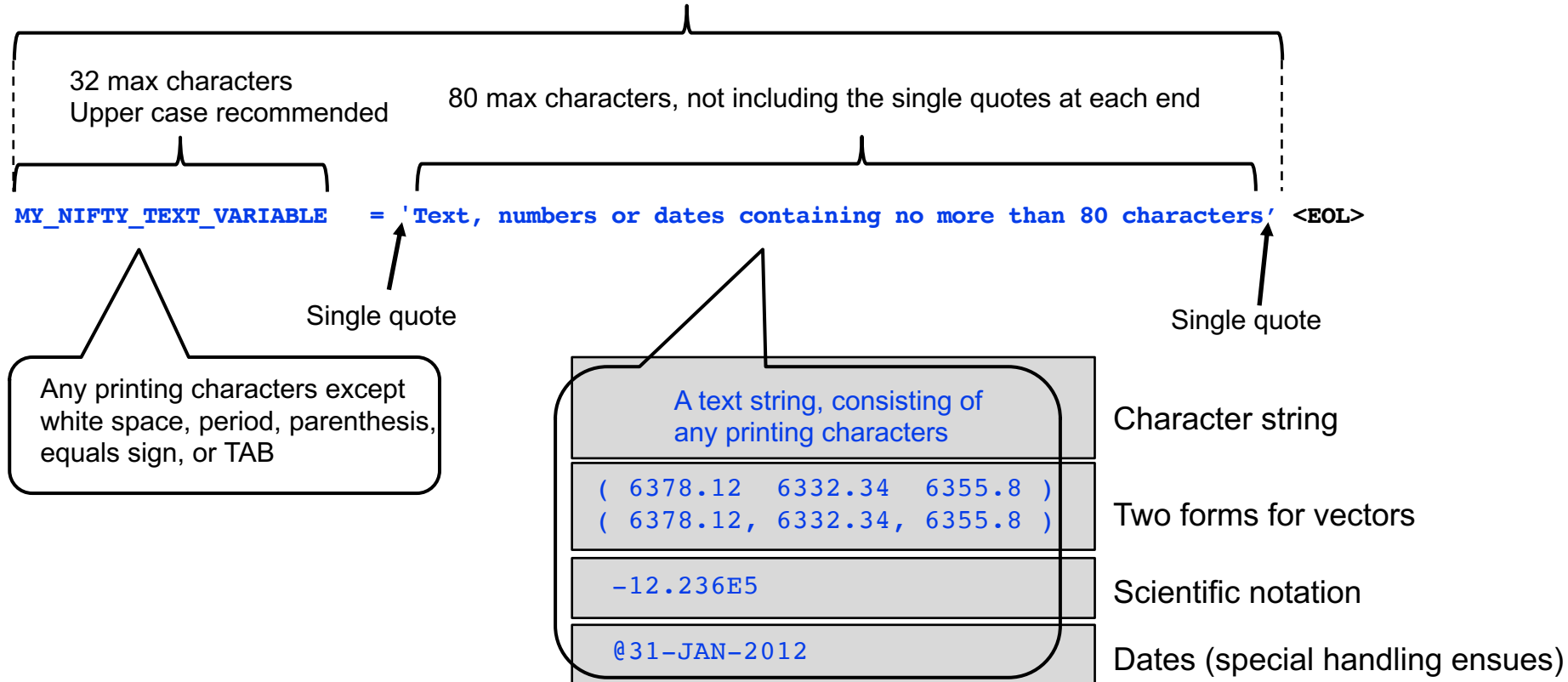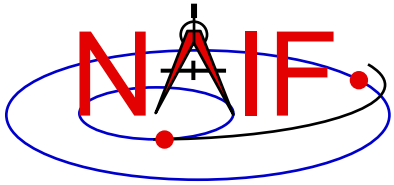- **See "Kernel Required Reading" for more information**

- ## A "picture" of the most basic text kernel assignment rules

132 max characters, with the non-printing system-dependent end-of-line indicator at the end*

32 max characters
Upper case recommended

80 max characters, not including the single quotes at each end

`MY_NIFTY_TEXT_VARIABLE  = 'Text, numbers or dates containing no more than 80 characters' <EOL>`

Single quote

Single quote

Any printing characters except white space, period, parenthesis, equals sign, or TAB

| | |
|---|---|
| A text string, consisting of any printing characters | Character string |
| ( 6378.12  6332.34  6355.8 )<br>( 6378.12, 6332.34, 6355.8 ) | Two forms for vectors |
| −12.236E5 | Scientific notation |
| @31−JAN−2012 | Dates (special handling ensues) |

*Unix, Linux, OSX EOL symbol:   \<LF>
*DOS/Windows EOL symbol:    \<CR>\<LF>

**Navigation and Ancillary Information Facility**

A binary kernel contains lots of non-printing data, usually interspersed with occasional occurrences of ASCII characters.

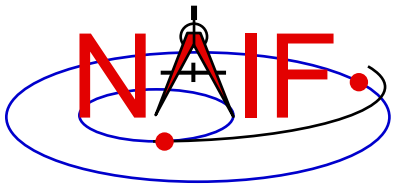Includes a "comment area" where descriptive meta-data should be placed.
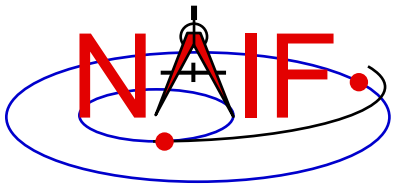
- **All SPICE kernels should contain comments– descriptive information about the data contained in the file.**
  - **"Comments" are also known as "meta-data"**

- **See the tutorial on comments for more information.**

# Making a Text Kernel

- **Text kernels may be produced using a text editor**
  - Text kernels must contain only printing characters (ASCII values 32-126), i.e. human-readable text
    » TAB characters are allowed but HIGHLY DISCOURAGED
    » Caution: some editors insert non-printing characters
  - Text kernels must have each line terminated with the end-of-line indicator appropriate for the operating system you are using
    » For Unix, PC/Linux, Mac OSX:   <LF>
    » For PC/Windows:                    <CR><LF>
    » **Don't forget to insert the end-of-line indicator on the very last line of the kernel!**
  - Fortran toolkit software will detect and warn you if you try to read a non-native text kernel. (Not needed for other languages.)
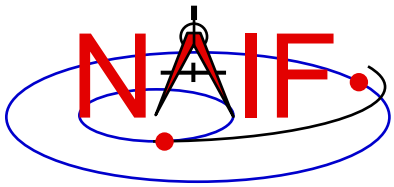  - See the BACKUP for information on converting text kernels between these two line termination techniques
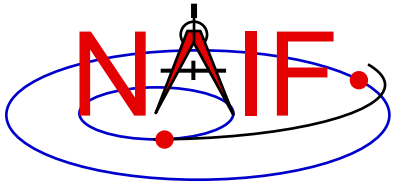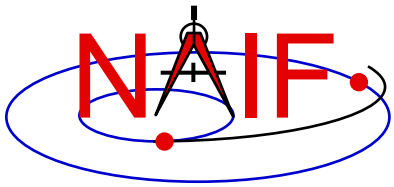
# Using Kernels

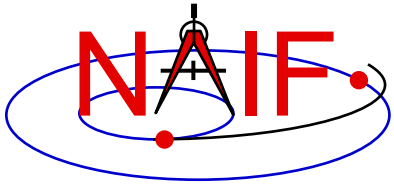**Navigation and Ancillary Information Facility**

- **To make kernels available to a program you "load" them**

- **When you load a <u>text</u> kernel:**
  - **the file is opened**
  - **the kernel contents are read into memory**
    - » **variable names and associated values are stored in a data structure called the "kernel pool"**
  - **the file is closed**

- **When you load a <u>binary</u> kernel:**
  - **the file is opened**
  - **for SPK, CK, binary PCK and DSK files, no data are read until a read request is made by Toolkit software**
  - **for all practical purposes the binary file remains open unless specifically unloaded by you**

**Navigation and Ancillary Information Facility**

- **Use the FURNSH routine to load all kernels – text and binary**
  - `CALL FURNSH ( 'name.ext' )` **(Fortran)**
  - `furnsh_c ( "name.ext" );` **(C)**
  - `cspice_furnsh, 'name.ext'` **(IDL)**
  - `cspice_furnsh ( 'name.ext' )` **(MATLAB)**
  - `spiceypy.furnsh ( 'name.ext' )` **(Python using SpiceyPy)**

- **Best practice: don't hard code filenames–list these in a "meta-kernel" and load the meta-kernel using FURNSH**
  - `CALL FURNSH ( 'meta-kernel_name' )` **(Fortran example)**
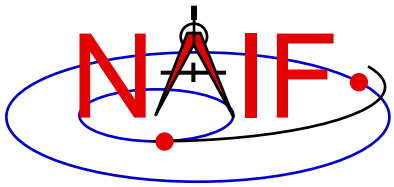  - **Look further down for more information on meta-kernels**

# Meta-Kernels

# These help make using SPICE easy!

# What is a "Meta-Kernel"

- **A meta-kernel is a file that lists names (and locations) of a collection of SPICE kernels that are to be used together in a SPICE-based application**
  - Loading the meta-kernel causes all of the kernels listed in it to be loaded

- **Using a meta-kernel makes it easy to manage which SPICE files are loaded into your program–you don't need to revise your code**

- **A meta-kernel is implemented using the SPICE text kernel standards**
  - Refer to the Kernel Required Reading technical reference for details

- **The terms "meta-kernel" and "FURNSH kernel" are used synonymously**
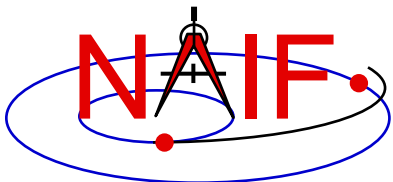
**Navigation and Ancillary Information Facility**

```
KPL/MK
\begindata
    KERNELS_TO_LOAD = (
                '/home/mydir/kernels/lowest_priority.bsp',
                '/home/mydir/kernels/next_priority.bsp',
                '/home/mydir/kernels/highest_priority.bsp',
                '/home/mydir/kernels/leapseconds.tls',
                '/home/mydir/kernels/sclk.tsc',
                '/home/mydir/kernels/c-kernel.bc',
                '/home/mydir/kernels+',
                '/custom/kernel_data/p_constants.tpc',
                         )
```

All the commas
are optional

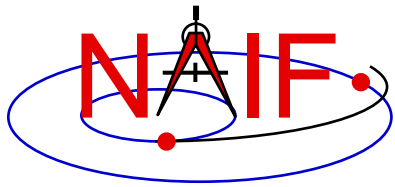# Sample Meta-Kernel

```
KPL/MK
\begindata
   KERNELS_TO_LOAD = (
               '/home/mydir/kernels/lowest_priority.bsp',
               '/home/mydir/kernels/next_priority.bsp',
               '/home/mydir/kernels/highest_priority.bsp',
               '/home/mydir/kernels/leapseconds.tls',
               '/home/mydir/kernels/sclk.tsc',
               '/home/mydir/kernels/c-kernel.bc',
               '/home/mydir/kernels+',
               '/custom/kernel_data/p_constants.tpc',
                     )
```

All the commas are optional

• **The last file listed in this example (p_constants.tpc) demonstrates how to use the continuation character, '+', to work around the 80 character limitation imposed on string lengths by the text kernel standards.**

• **See the next two pages for some important OS-specific details!**

**Navigation and Ancillary Information Facility**

• **This meta-kernel uses the PATH_VALUES and PATH_SYMBOLS keywords to specify the directory where the kernels are located.**

```
KPL/MK
\begindata
   PATH_VALUES     = ( '/home/mydir/kernels' )
   PATH_SYMBOLS    = ( 'KERNELS'             )
   KERNELS_TO_LOAD = (
          '$KERNELS/lowest_priority.bsp',
          '$KERNELS/next_priority.bsp',
          '$KERNELS/highest_priority.bsp',
          '$KERNELS/leapseconds.tls',
          '$KERNELS/sclk.tsc',
          '$KERNELS/c-kernel.bc',
          '$KERNELS/custom/kernel_data/p_constants.tpc'
                     )
```

**UNIX/MAC style path notation, using forward slashes**

• **Although the OS environment variable notation $<name> is used to refer to the symbols specified using the PATH_VALUES and PATH_SYMBOLS keywords, these symbols are NOT operating system environment variables and are set and used for substitution by SPICE only in the context of this particular meta-kernel.**
• **The '+' continuation character described on the previous page may be used to handle path strings that exceed 80 characters.**

**Navigation and Ancillary Information Facility**

• **This meta-kernel uses the PATH_VALUES and PATH_SYMBOLS keywords to specify the directory where the kernels are located.**
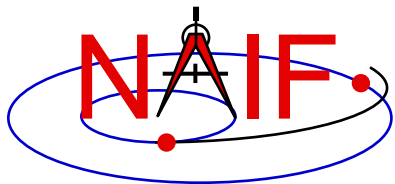
```
KPL/MK
\begindata
    PATH_VALUES     = ( 'c:\home\mydir\kernels' )
    PATH_SYMBOLS    = ( 'KERNELS'              )
    KERNELS_TO_LOAD = (
            '$KERNELS\lowest_priority.bsp',
            '$KERNELS\next_priority.bsp',
            '$KERNELS\highest_priority.bsp',
            '$KERNELS\leapseconds.tls',
            '$KERNELS\sclk.tsc',
            '$KERNELS\c-kernel.bc',
            '$KERNELS\custom\kernel_data\p_constants.tpc'
                    )
```

**Windows style path notation, using backwards slashes**

• **Although the OS environment variable notation $<name> is used to refer to the symbols specified using the PATH_VALUES and PATH_SYMBOLS keywords, these symbols are <u>NOT</u> operating system environment variables and are set and used for substitution by SPICE only in the context of this particular meta-kernel.**
• **The '+' continuation character described previously may be used to handle path strings that exceed 80 characters.**
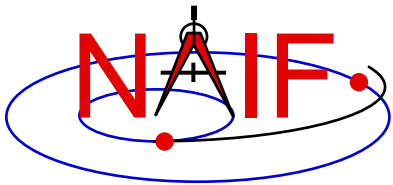
# Kernel Precedence Rule

- **The order in which SPICE kernels are loaded at run-time determines their priority when requests for data are made**

  - **For binary kernels, data from a higher priority file will be used in the case when two or more files contain data overlapping in time for a given object.**

    - » **For SPKs, CKs, and binary PCKs the file loaded <span style="color:red">last</span> takes precedence (has higher priority).**

    - » **For DSKs, use of priority will be specified via API calls**

      - » **Not yet supported as of N66 Toolkits**

    - » **Priority doesn't apply to ESQ files – all data from all loaded files are available.**

  - **If two (or more) text kernels assign value(s) to a single keyword using the "=" operator, the data value(s) associated with the last loaded occurrence of the keyword are used–all earlier values are replaced with the last loaded value(s).**

  - **Orientation data from a binary PCK <u>always</u> supersedes orientation data (for the same object) obtained from a text PCK, no matter the order in which the kernels are loaded.**
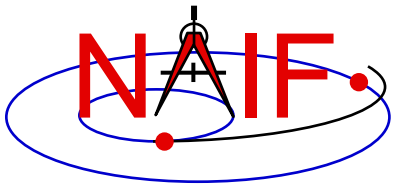
# Unloading Kernels

- **The unloading of a kernel is infrequently needed for FORTRAN or CSPICE applications but is essential for Icy, Mice, Python and similar interpreter scripts.**
  - Because of the way IDL and MATLAB interact with external shared object libraries, any kernels loaded during an IDL or MATLAB session will stay loaded until the end of the session unless they are specifically unloaded.

- **The routines KCLEAR and UNLOAD may be used to unload kernels containing data you wish to be no longer available to your program.**
  - KCLEAR unloads all kernels and clears the kernel pool
  - UNLOAD unloads specified kernels
  - KCLEAR and UNLOAD are only capable of unloading kernels that have been loaded with the routine FURNSH. They will not unload any files that have been loaded with older load routines such as SPKLEF (those used prior to availability of FURNSH).

- **Caution: unloading text kernels with UNLOAD will also remove any kernel pool data provided through the kernel pool run-time data insertion/update APIs (PCPOOL, PDPOOL, PIPOOL).**