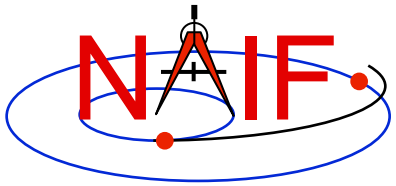
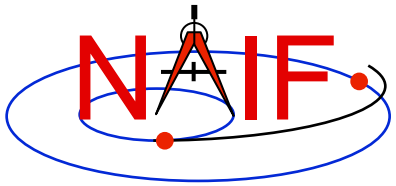


Navigation and Ancillary Information Facility

Introduction to Kernels

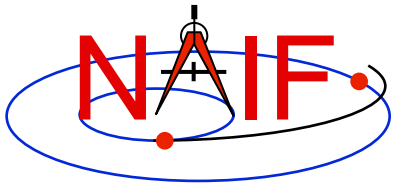
April 2016



Agenda

Navigation and Ancillary Information Facility

- Overview
- Kernel architecture
- Producing kernels
- Using kernels



What is a SPICE “Kernel”

Navigation and Ancillary Information Facility

“Kernel” means file

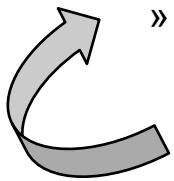
“Kernel” means a file containing ancillary data

“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

Navigation and Ancillary Information Facility

- **SPK**
  - Spacecraft and Planet Ephemeris
- **PCK**
  - Planetary Constants, for natural bodies
    - » Orientation
    - » Size and shape
- **IK**
  - Instrument
- **CK**
  - Pointing ("C-matrix")
- **EK**
  - 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
- **DBK**
  - Database mechanism
    - » Primarily used to support the ESQ

Text and Binary Kernels

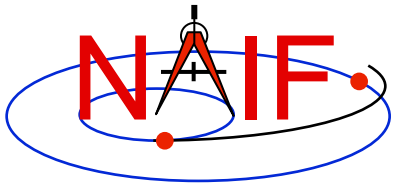
SPICE text kernels are:

- 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



SPICE Kernel Forms

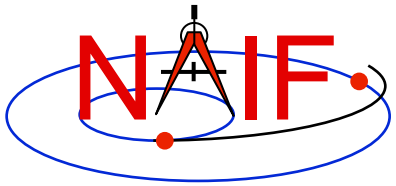
Navigation and Ancillary Information Facility

- **Binary form: SPK, binary PCK, CK, EK/ESQ[1], DSK**
  - A file mostly containing data encoded in binary form
  - Binary kernels are not human-readable; they require the use of Toolkit software to examine the data contents
- **Text form: text PCK, IK, FK, LSK, SCLK, MK**
  - Plain text files containing only printing characters (ASCII values 32-126), i.e. human-readable text.
- **"Transfer" form of a binary kernel**
  - This is an ASCII representation of a binary kernel
  - Was used for porting the file between computers with incompatible binary representations (e.g. PC and UNIX)
  - Use of the transfer format is no longer needed for porting due to the run-time translation capability added to SPICE long ago
    - » But it is one way to convert a non-native binary kernel into native format, needed for modifying the kernel or improving read efficiency

[1] The ESP and ENB components of the EK might be binary, or text, or html, depending on specific implementation.

Kernel Architecture

- Text kernels
- Binary kernels
- Comments in kernels

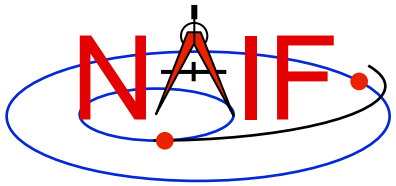


Text Kernel Contents

- 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

Navigation and Ancillary Information Facility

```
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

   \begintext

      < some comments about the data >

   \begindata

      < more data in keyword = value syntax >

   \begintext
         < etc., etc. >
```

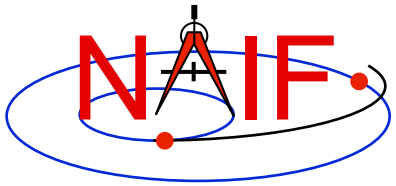
A data block

A “comments” block

Another data block

Another “comments” block

- **The next several pages describe what you see above**
- **See the “Kernel Required Reading” document for details**



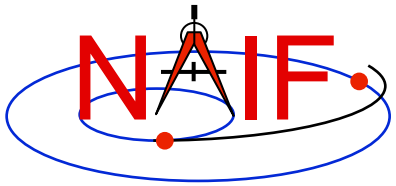
Text Kernel Formatting

- `KPL/<kernel type>`
  - Appears on the first line
  - Tells SPICE software what kind of kernel it is
  - Examples of kernel type are FK, IK, PCK, SCLK

- `\begindata` and `\begintext`
  - Markers, on lines by themselves, which set off the beginning of data and the beginning of meta-data blocks respectively

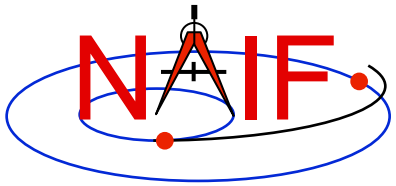
- `<LF>` for Unix/Linux/Mac or `<CR><LF>` for Windows
  - End of line marker (usually not visible when you display a text kernel)
  - Must be present on EVERY line in the text kernel

- Max line length, including any white space
  - 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 J2000
    - » There is no time system implied
    - » This conversion does not need a leap seconds kernel



Variable Names and Values

Navigation and Ancillary Information Facility

- **Variable names**
  - Max of 32 characters
  - Are case sensitive (recommendation: use only upper case)
  - Cannot include space, period, comma, 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 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**

Example Binary Kernel

Navigation and Ancillary Information Facility

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

Includes a “comment area” where descriptive meta-data should be placed.

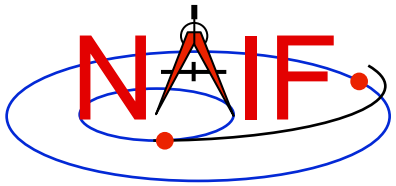
Comments In SPICE Kernels

Navigation and Ancillary Information Facility

- **All SPICE kernels should contain comments– descriptive information about the data contained in the file.**
  - **“Comments” are also known as “meta-data”**

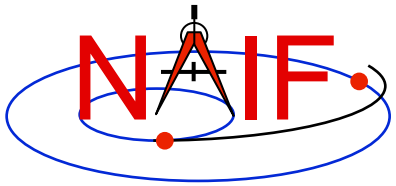
- **See the tutorial on comments for more information.**

Producing Kernels



Making a Text Kernel

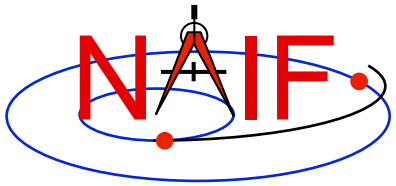
- **Text kernels may be produced using any available text editor**
  - **Remember, text kernels must contain only printing characters (ASCII values 32-126), i.e. human-readable text**
    - » **TAB characters are allowed but HIGHLY DISCOURAGED**
  - **Text kernels must have each line terminated with 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!**
  - **Toolkit software will detect and warn you if trying to read a non-native text kernel**
    - » **Caution: this warning doesn't work for a file smaller than 132 bytes**
  - **See the BACKUP for information on converting text kernels between these two line termination techniques**



Making a Binary Kernel

- Binary kernels are made using Toolkit utility programs, or by using Toolkit APIs built into your own application program
- See “How Kernels are Made and Used” in the BACKUP section for a bit more information
- See the “Making an SPK” and “Making a CK” tutorials

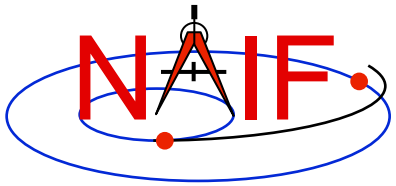
Using Kernels



Loading Kernels - 1

Navigation and Ancillary Information Facility

- To make kernels available to a program you “load” them
- When you load a text 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 binary kernel:
  - the file is opened
  - for SPK, CK, and binary PCK files, no data are read until a read request is made by Toolkit software
  - for ESQ files, the schema description is read, checked, and stored in memory at load time, but no data are read until a query/fetch is made
  - for all practical purposes the binary file remains open unless specifically unloaded by you

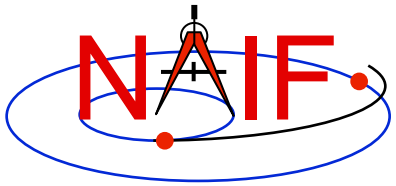


Loading Kernels - 2

- **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)

- **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)
  - **See the next page for more information on meta-kernels**

- **Caution: "Transfer format" versions of binary kernels can not be loaded; they must first be converted to binary with the Toolkit utility program *tobin* or *spacit***

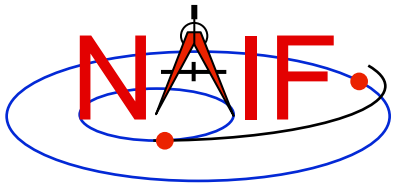


Run-time Translation

- **Binary kernels, whether or not in native binary format, may be read by any of the toolkits**
  - **Accomplished by run-time translation built into Toolkit code**
  - **Run-time translation does NOT apply to writing to an existing binary kernel**

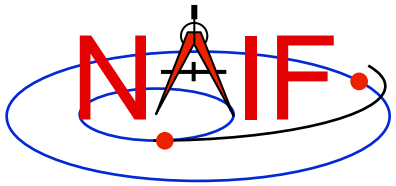
- **Text kernels may be read by any of the C, IDL and Matlab Toolkits**
  - **For these, run-time translation is accomplished by Toolkit code**
  - **Run-time translation does NOT work for Fortran Toolkits: these Toolkits read text kernels only in native format**

Meta-Kernels



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
  - You can simply load the meta-kernel, causing 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
- 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



Sample Meta-Kernel

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',
                        )
```

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 sizes by the text kernel standards.

- See the next two pages for some important details!

Unix/Mac Sample Meta-Kernel

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 (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.

Windows
Sample Meta-Kernel

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'
                      )
```

Windows style path notation (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 NOT operating system environment variables and are set and used for substitution by SPICE only in the context of this particular meta-kernel.

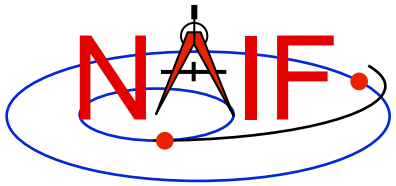
• The '+' continuation character described previously may be used to handle path strings that exceed 80 characters.

Limits on Loaded Kernels (N65)

Navigation and Ancillary Information Facility

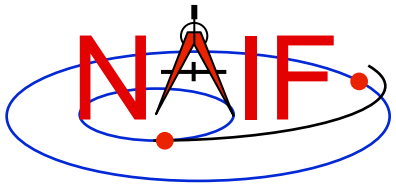
- **The number of binary kernels that may be loaded at any time is large, but limited.**
  - **For SPK, CK, and binary PCK files:**
    - » **Loaded SPKs + Loaded CKs + Loaded binary PCKs <= 5000**
  - **For ESQ files:**
    - » **Loaded ESQs <= 20**
  - **For all kernels:**
    - » **Loaded kernels <= 5000**
      - **Assumes each has been loaded only once, and not unloaded.**
- **There are also limits on the number of keywords and values for all loaded text kernels:**
  - **Maximum number of keywords is 26,003**
  - **Maximum number of numeric data items is 400,000**
  - **Maximum number of character data items is 15,000**

Some increases will be made in the N66 Toolkits soon to be released.



Kernel Precedence Rule

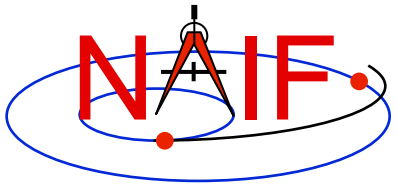
- **The order in which SPICE kernels are loaded at run-time determines their priority when requests for data are made**
  - **– For binary kernels, data from a higher priority file will be used in the case when two or more files contain data overlapping in time for a given object.**
    - **» For SPKs, CKs and binary PCKs the file loaded last takes precedence (has higher priority).**
    - **» 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 have been replaced with the last loaded value(s).**
  - **– Orientation data from a binary PCK always supersedes orientation data (for the same object) obtained from a text PCK, no matter the order in which the kernels are loaded.**



Unloading Kernels

Navigation and Ancillary Information Facility

- The unloading of a kernel is infrequently needed for FORTRAN or CSPICE applications but is essential for Icy and Mice 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 APIs (PCPOOL, PDPOOL, PIPOOL)

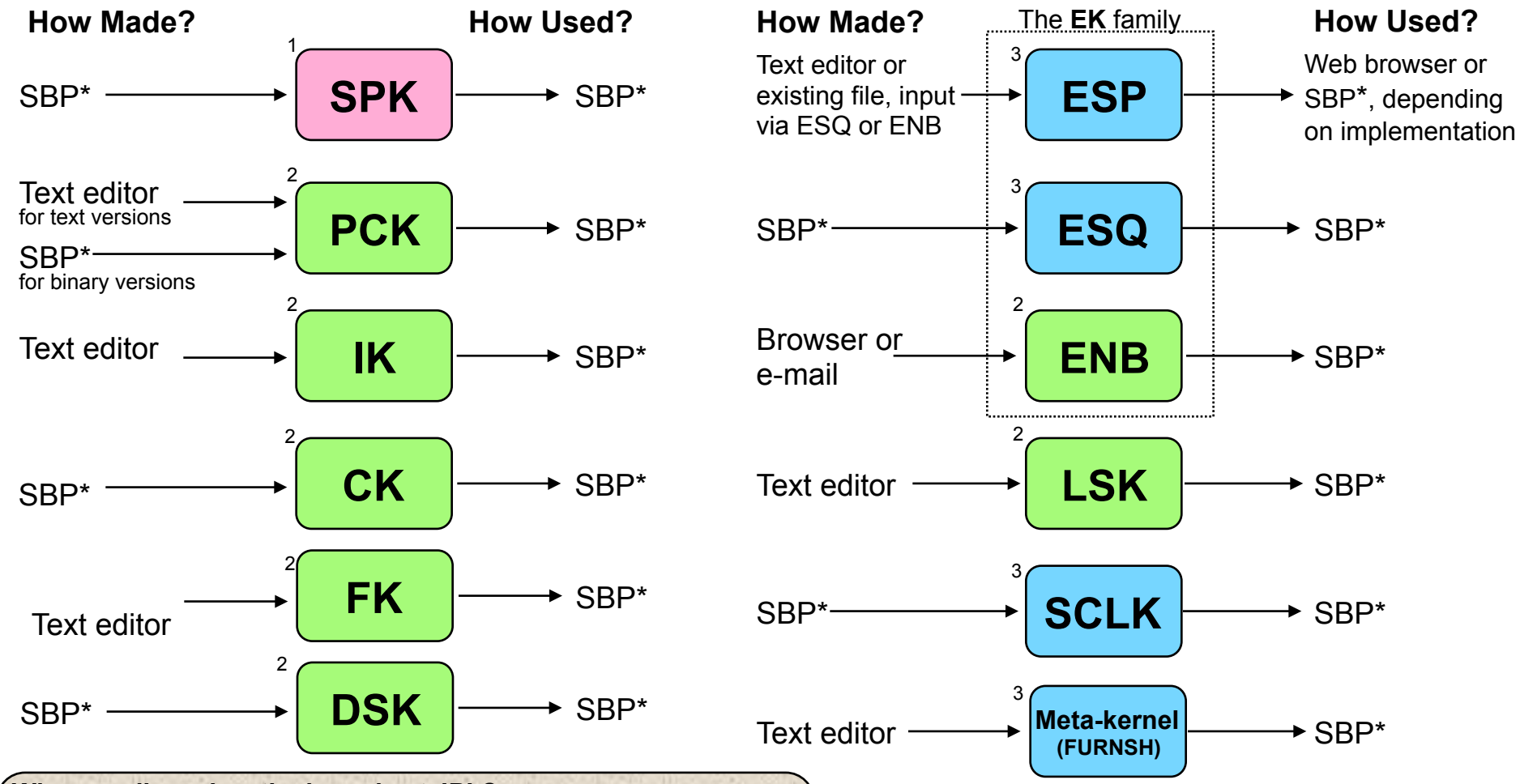


Backup

- How kernels are made and used
- Why and how kernels are modified
- SPICE data structures hierarchy
- Problems making text kernels

How Kernels are Made and Used at JPL

Navigation and Ancillary Information Facility



Who usually makes the kernels at JPL?

1 **NAV and NAIF**

2 **NAIF**

3 **NAIF or other**

This represents current practice for most JPL missions, but is by no means a requirement. Anyone can make SPICE files.

***SBP** = SPICE-based program that uses modules from the SPICE Toolkit. In some cases the Toolkit contains such a program already built. In some cases NAIF may have such a ready-built program that is not in the SPICE Toolkit.

Why & How Kernels are "Modified" - 1

Navigation and Ancillary Information Facility

File Type	Why Modified	How Modified
SPK	-To add comments -To merge files or subset a file -To correct/revise an object ID	- COMMNT, SPACIT or SPICELIB module - SPKMERGE - BSPIDMOD
PCK **Text version**	-To revise data values -To add additional data items and values	- Text editor - Text editor
IK	-To revise data values -To add additional data items and values	- Text editor - Text editor
CK	-To add comments -To merge files -To revise the interpolation interval -To subset a file	- COMMNT, SPACIT, or SPICELIB module - DAFCAT, CKSMRG - CKSPANIT, CKSMRG - CKSLICER
FK	-To revise data values -To add additional data items and values	- Text editor - Text editor
DSK	-To add comments -To merge files or subset a file	- COMMNT, SPACIT or SPICELIB module - DSKMERGE

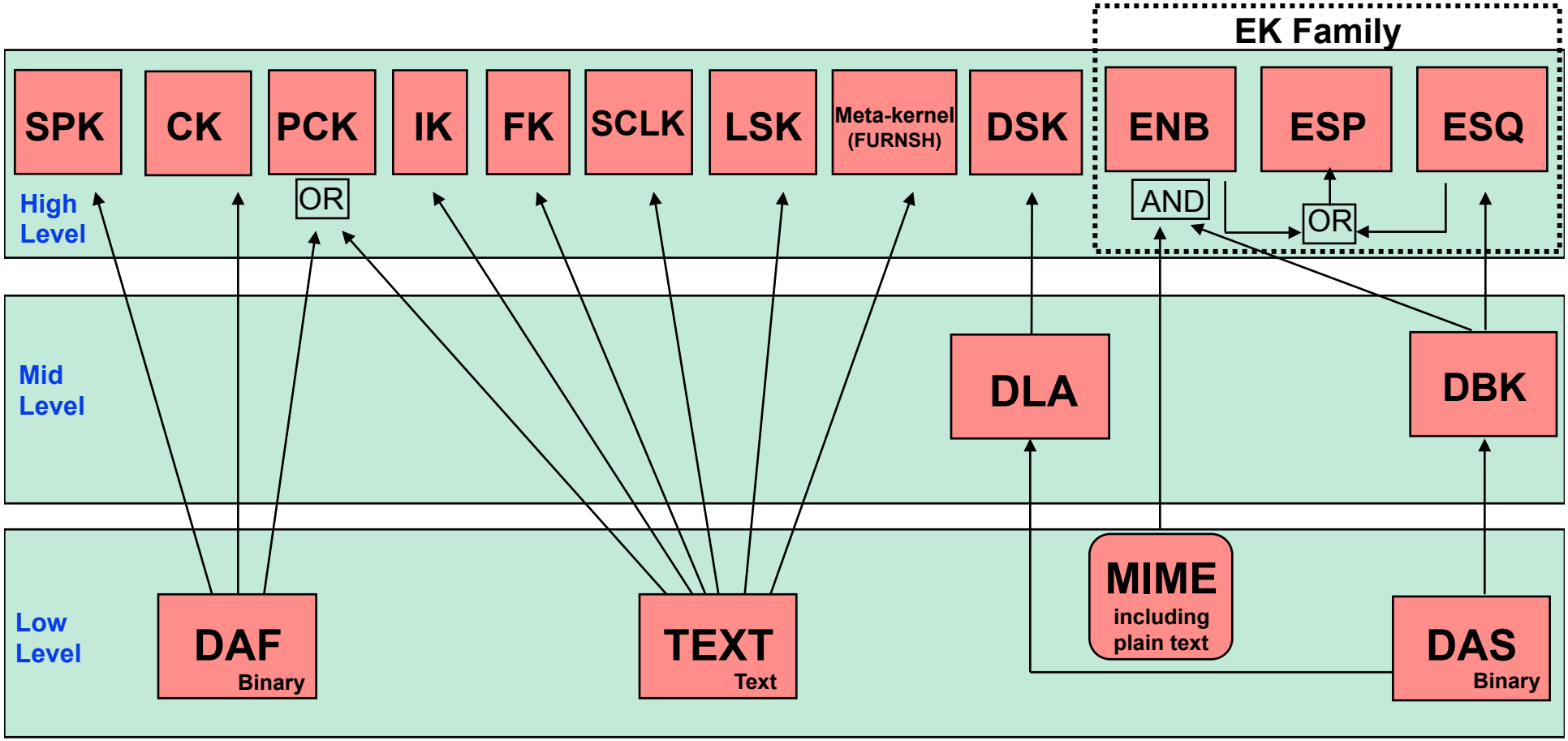
Why & How Kernels are "Modified" - 2

Navigation and Ancillary Information Facility

File Type	Why Modified	How Modified
The **EK** family		
ESP	-To add, revise or delete "data" -To add comments	- (Depends on implementation) - (Depends on implementation)
ESQ	-To add additional data -To revise data -To delete data -To add comments -To merge files	- Toolkit modules - Toolkit modules - Toolkit modules - COMMNT, SPACIT or SPICELIB module - (under development)
ENB	-To change entry status (public <--> private) -To delete an entry	- WWW - WWW
LSK	- To add a new leapsecond	- Text editor
SCLK	- To add comments	- Text editor
Meta-kernel (FURNSH)	- To revise contents in any way	- Text editor

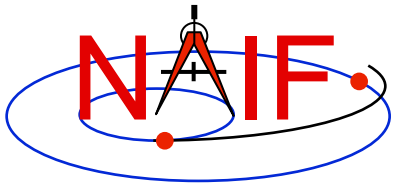
SPICE Data Structures Hierarchy

Navigation and Ancillary Information Facility



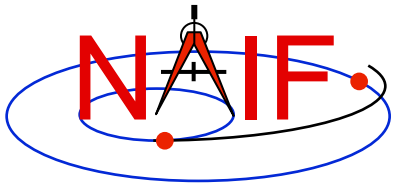
DAF = Double Precision Array File
DBK = Data Base Kernel
DAS = Direct Access, Segregated
DSK = Digital Shape Kernel (under development)
DLA = DAS Linked Array (under development)

Excepting MIME, each of these data structures is built entirely of SPICE components.
PCK files are usually text-based, but binary versions exist for the earth and moon. The ESP has been implemented using both the ENB and ESQ mechanisms. The DBK is a SQL-like, homebrew database.



Problems Making Text Kernels

- **Cutting/pasting complete, or pieces of, data assignments or `\begindata` or `\begintext` markers into a text kernel can cause a problem**
  - **It may result in insertion of non-printing characters or incorrect end-of-line terminations**
  - **This is not a problem for comments, but it is probably best to treat all portions of a text kernel the same**
- **If creating a text kernel by editing an existing one:**
  - **first save a backup copy**
  - **be sure you are starting with a file in native format for the computer you are using: either Unix/Linux/Mac or Windows**
  - **be sure to insert a final end-of-line marker at the end of your last line of data or text**
    - » **Press “return”**

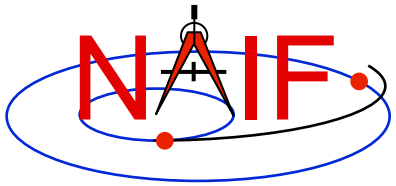


Some Useful Tools - 1

- **For a Unix or Linux (including Mac) environment**
  - **To display all non-printing characters, display tab characters as "^I", and place a "$" character at the end of each line**
    - » `cat -et <file name>`
    - » **How do end-of-line markers appear when displayed in a text file using the cat -et command?**
      - **Unix/Linux/Mac: $ (line feed)**
      - **Windows: ^M$ (carriage return followed by line feed)**
  - **Display the file type, language used, and end of line marker**
    - » `file <file name>`
    - » **Examples using Unix and Windows ("PC") versions of the SPICE leapseconds kernel:**
      - `file naif0010.tls`

        `naif0010.tls: ASCII English text`
      - `file naif0010.tls.pc`

        `naif0010.tls.pc: ASCII English text, with CRLF line terminators`



Some Useful Tools - 2

- **For a Unix or Linux (including Mac) environment**
  - **To convert a Unix/Linux/Mac text kernel to Windows ("DOS") style:**
    - » **unix2dos <filename>**
  - **To convert a Windows ("DOS") style text kernel to Unix/Linux/Mac style:**
    - » **dos2unix <filename>**
  - **For converting either style text kernel to the other style, use the SPICE "bingo" executable**
    - » **The bingo program and User Guide are available only from the NAIF/Toolkit/Utilities web page:**
      - **http://naif.jpl.nasa.gov/naif/utilities.html**
- **More information**

  In Wikipedia, search on "newline" or "unix2dos"