Non-Toolkit Applications

January 2017

Summary

- NAIF makes available a set of applications *not* included in the generic Toolkits. This set includes programs for:
  - making, modifying, validating, inspecting, and analyzing SPK files:
    - » *pinpoint, dafcat, bspidmod, dafmod, spy*
  - making and modifying CK files
    - » *prediCkt, ckslicer, ckspanit, dafcat, cksmrg, dafmod*
  - making SCLK files
    - » *makclk*
  - computing derived quantities
    - » *orbnum, optics, spy*
  - determining SPICE kernel type and binary format
    - » *archtype, bff*
  - converting between binary and text kernel formats
    - » *bingo*
- Executables and User's Guides for selected computer environments are available from the NAIF server at:
  - http://naif.jpl.nasa.gov/naif/utilities.html

Using Non-Toolkit Apps

- **All of these apps are meant to be used as operating system shell executables**
  - **One generally cannot run these within IDL or MATLAB– run them from an operating system shell.**
    - **In some cases you can run from within IDL or MATLAB, but this is not recommended:**
      - **In IDL, use the "spawn" command**
      - **In MATLAB, use the "system" command**

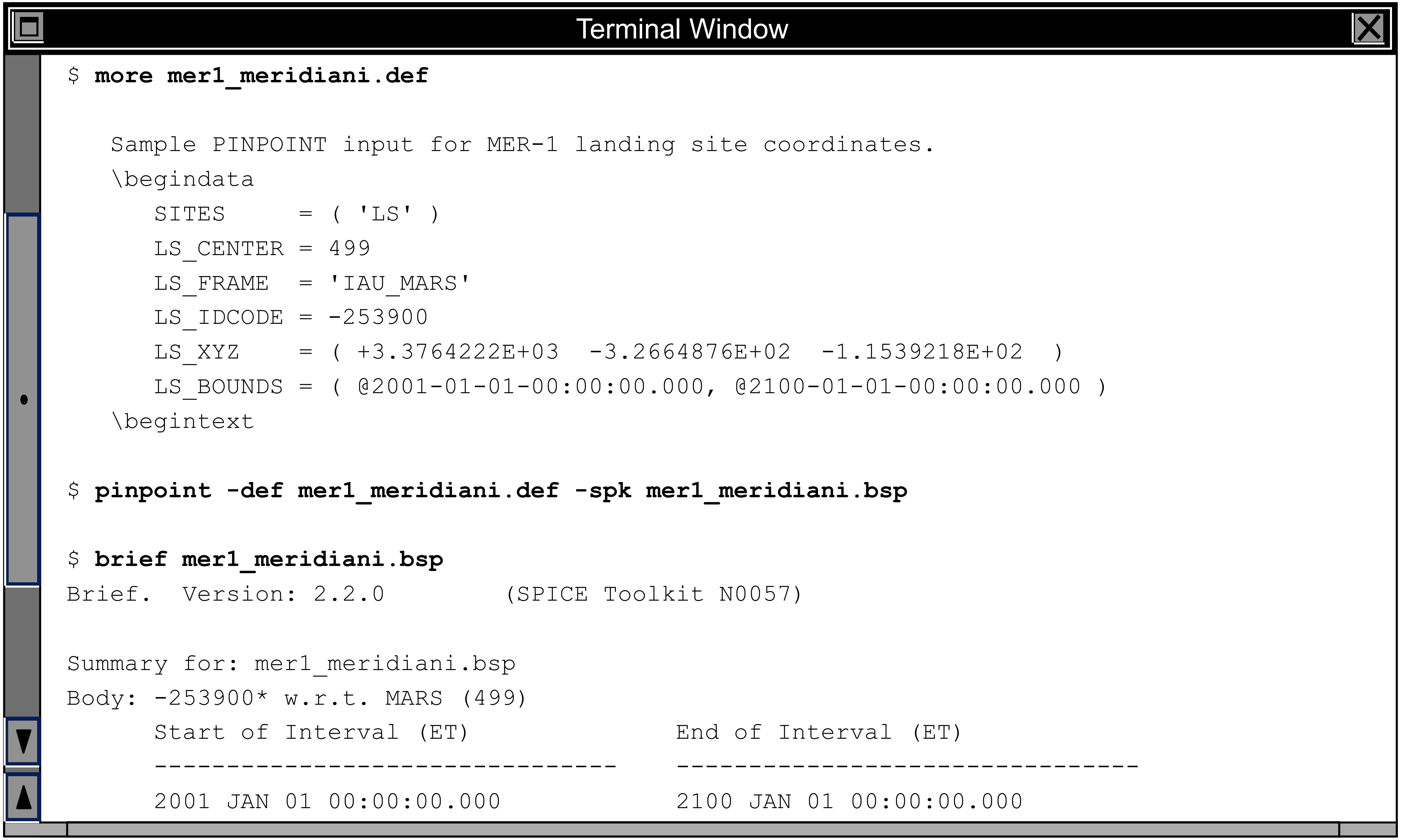
PINPOINT

- *pinpoint* is a program for creating SPK files and topocentric frames FK files for objects whose position is a constant offset with respect to another object
  - Ground stations
  - Landing sites, sites along a rover path
  - Relative positions of manipulator joints, etc.
- *pinpoint* is a command line program with the following usage:

```
pinpoint -def deffile -spk spkfile [-pck tkfile] [-fk fk] [flags]
```

  - “deffile” is an input definitions file following text kernel file format and containing a set of keywords defining ID, center, reference frame, position (as XYZ or Gaussian Lat/Lon/Alt) and time coverage boundaries, and optionally velocity and topocentric frame axes specifications, for one or more objects
    - The contents of “deffile” are included in the comment area
  - “spkfile” is an output SPK file containing a type 8 SPK segment for each of the objects specified in the “deffile”
  - “tkfile” is an optional input PCK file (needed if positions in the “deffile” are given as Lat/Lon/Alt) or FK file (needed if one or more of the frames specified in “deffile” is not one of the frames built into the Toolkit)
  - “fk” is an optional output topocentric frames FK file

PINPOINT Example



```
$ more mer1_meridiani.def

   Sample PINPOINT input for MER-1 landing site coordinates.
   \begindata
      SITES     = ( 'LS' )
      LS_CENTER = 499
      LS_FRAME  = 'IAU_MARS'
      LS_IDCODE = -253900
      LS_XYZ    = ( +3.3764222E+03  -3.2664876E+02  -1.1539218E+02  )
      LS_BOUNDS = ( @2001-01-01-00:00:00.000, @2100-01-01-00:00:00.000 )
   \begintext

$ pinpoint -def mer1_meridiani.def -spk mer1_meridiani.bsp

$ brief mer1_meridiani.bsp
Brief.  Version: 2.2.0        (SPICE Toolkit N0057)

Summary for: mer1_meridiani.bsp
Body: -253900* w.r.t. MARS (499)
      Start of Interval (ET)              End of Interval (ET)
      --------------------------------    --------------------------------
      2001 JAN 01 00:00:00.000            2100 JAN 01 00:00:00.000
```

DAFCAT

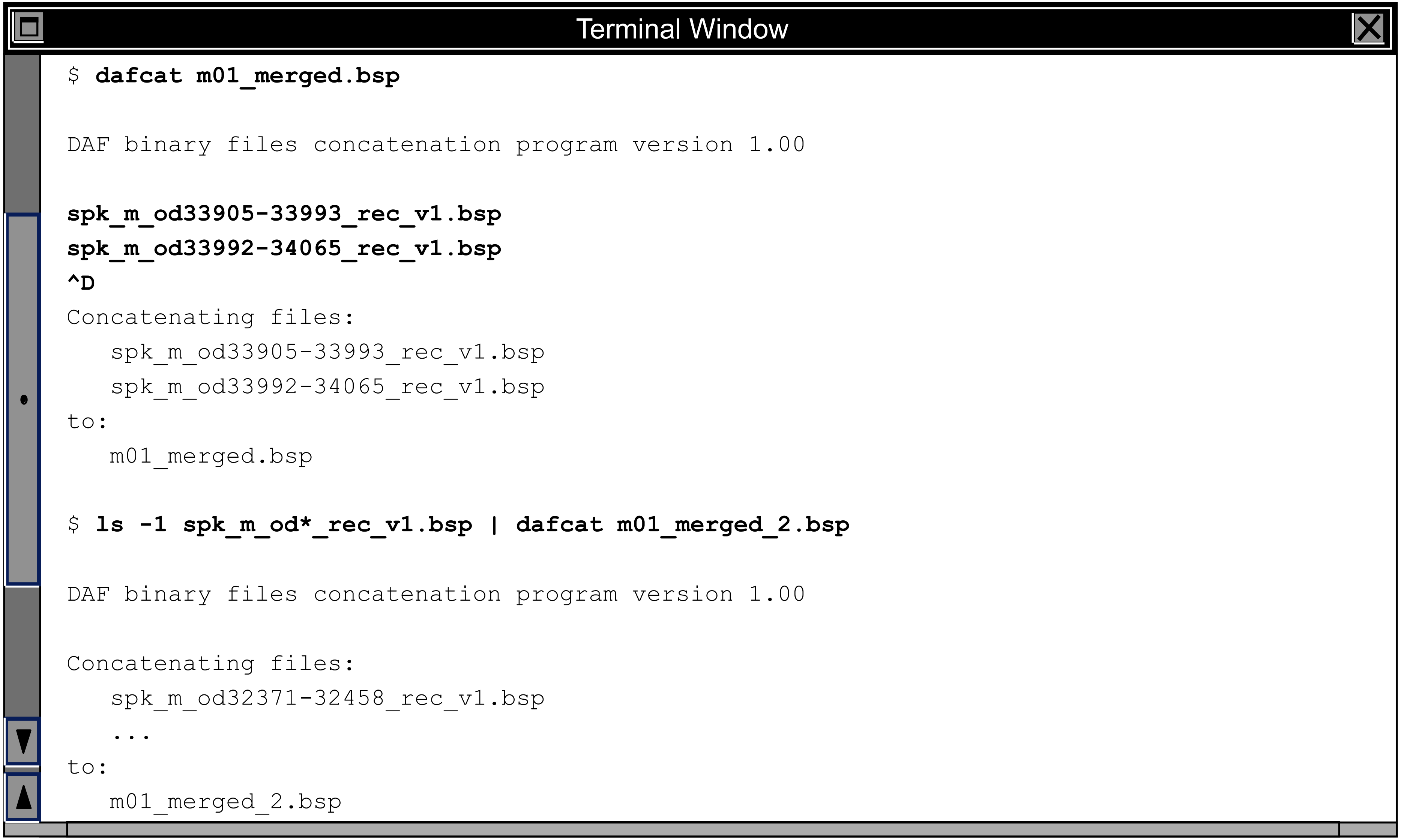
- *dafcat* is a program for concatenating binary DAF files by simply copying all data segments from all input files, in the order they are provided, into the output file
  - Works on SPKs, CKs, and binary PCKs
    - will not merge different types of kernels together, i.e. will not merge SPKs with CKs, CKs with PCKs, etc.
    - for merging SPKs in most cases *spkmerge* should be used instead because it provides a much more powerful and sophisticated capability
- *dafcat* is a command line program with the following usage

```
dafcat output_file
```

  - "output_file" is the output file name and is the program's only argument
  - Input file names are provided from standard input
    - this is very convenient for use with Unix shell pipes
- *dafcat* does not put any information into the comment area of the output file, leaving this responsibility to the user (use *commnt* to do so)

DAFCAT Example: SPK

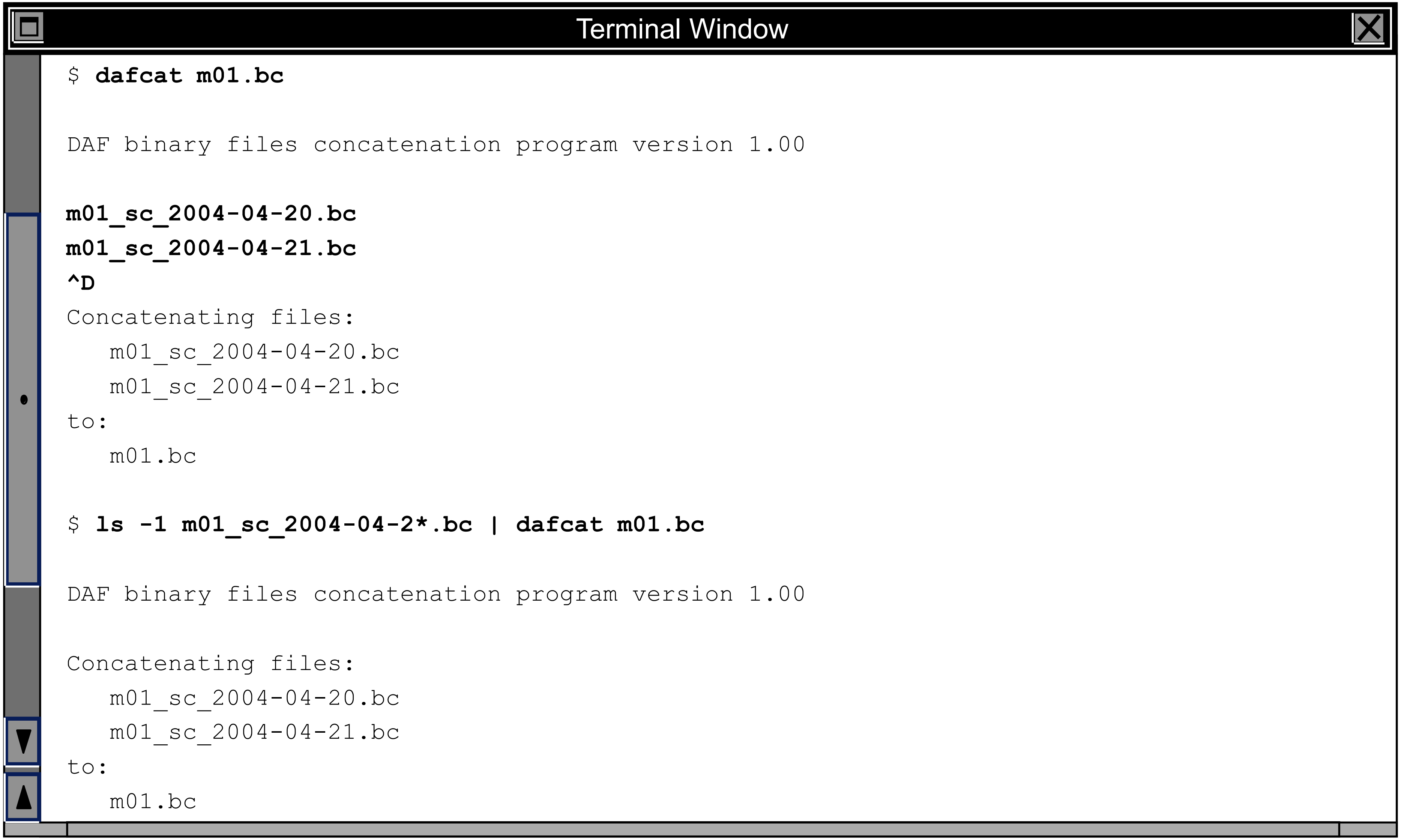
Navigation and Ancillary Information Facility



```
$ dafcat m01_merged.bsp

DAF binary files concatenation program version 1.00

spk_m_od33905-33993_rec_v1.bsp
spk_m_od33992-34065_rec_v1.bsp
^D
Concatenating files:
   spk_m_od33905-33993_rec_v1.bsp
   spk_m_od33992-34065_rec_v1.bsp
to:
   m01_merged.bsp

$ ls -1 spk_m_od*_rec_v1.bsp | dafcat m01_merged_2.bsp

DAF binary files concatenation program version 1.00

Concatenating files:
   spk_m_od32371-32458_rec_v1.bsp
   ...
to:
   m01_merged_2.bsp
```

DAFCAT Example: CK

Navigation and Ancillary Information Facility



```
$ dafcat m01.bc

DAF binary files concatenation program version 1.00

m01_sc_2004-04-20.bc
m01_sc_2004-04-21.bc
^D
Concatenating files:
   m01_sc_2004-04-20.bc
   m01_sc_2004-04-21.bc
to:
   m01.bc

$ ls -1 m01_sc_2004-04-2*.bc | dafcat m01.bc

DAF binary files concatenation program version 1.00

Concatenating files:
   m01_sc_2004-04-20.bc
   m01_sc_2004-04-21.bc
to:
   m01.bc
```

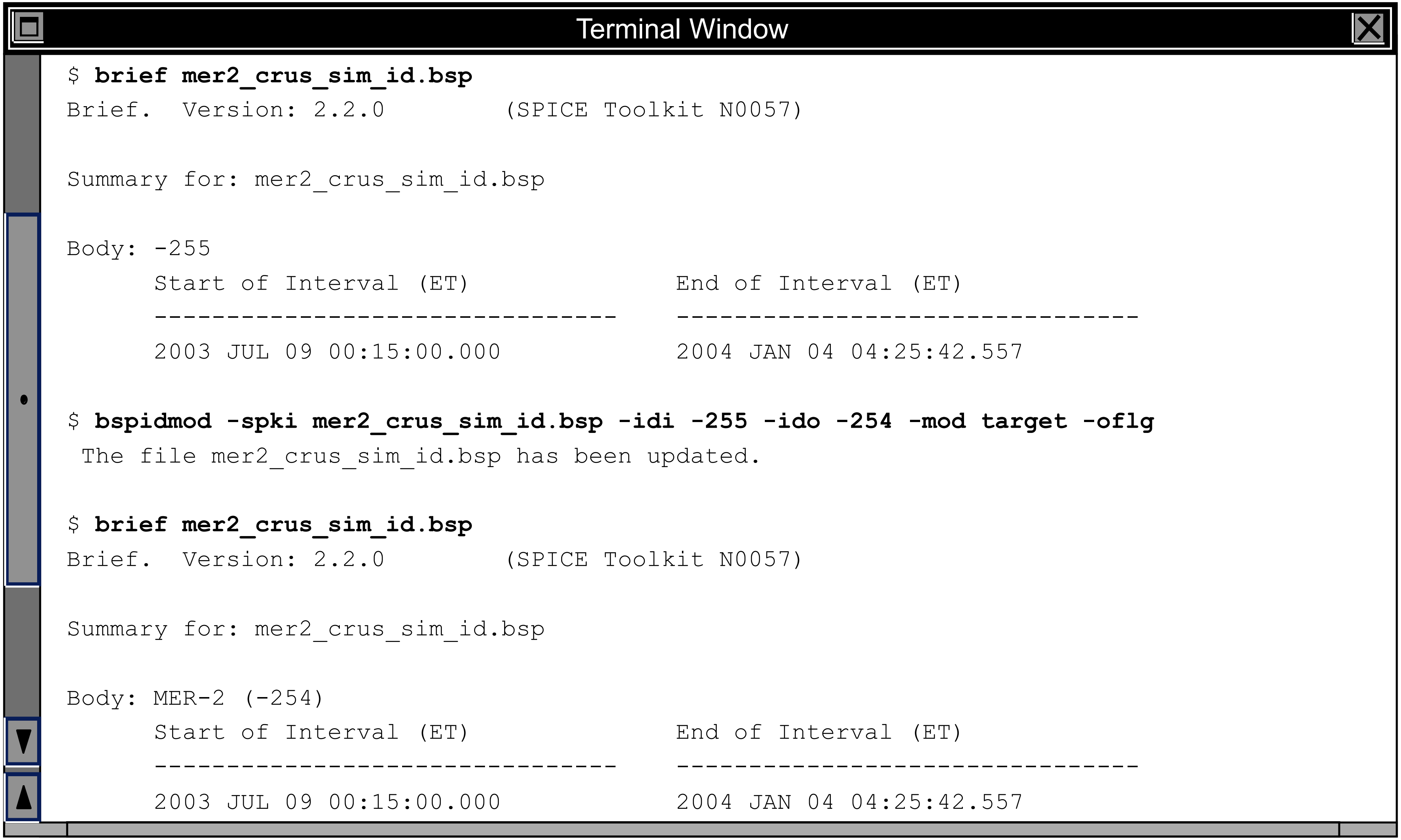
BSPIDMOD

- ***bspidmod*** **is a program for altering the object IDs in a binary SPK file**
  - can be used to modify IDs in an SPK file(s) produced with a “bogus” spacecraft ID (or a simulation spacecraft ID)
  - can be used to replace “good” IDs with “bogus” IDs if two different trajectories for the same object need to be used in the same program at the same time (for example for comparison)
- ***bspidmod*** **is a command line program with the following usage:**

```
bspidmod -spki inpspk -idi inpid -ido outid -mod item -oflg
```

  - “inpspk” is the input SPK file; “inpid” and “outid” are the current ID and new ID
  - “item” indicates which IDs are to be replaced:
    - TARGET -- only target IDs are replaced,
    - CENTER -- only center IDs are replaced, or
    - OBJECT -- both target and center IDs are replaced
    - » Replacements are made only when “inpid” matches an ID found in the input SPK
  - “-oflg” flag indicating that changes should be made directly in the input file; if not specified, the program produces an output file with name that has “_out” appended to the name of the input file
    - » In order for changes to be made in the input file it must be in the native binary format; if it is not, *bingo* may be used to convert it to the native binary format
  - A note stating which IDs were modified is put in the comment area

BSPIDMOD Example

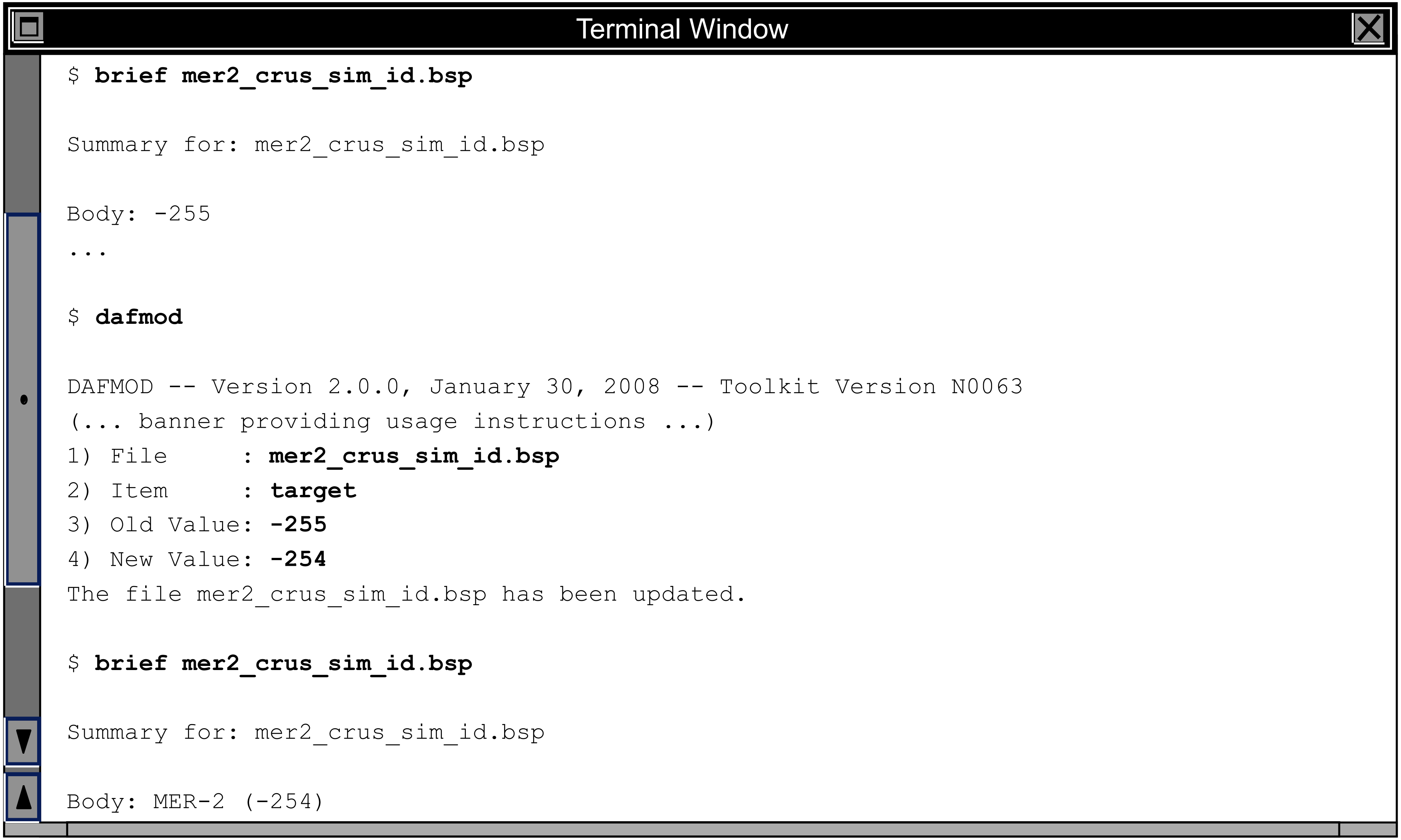


```
$ brief mer2_crus_sim_id.bsp
Brief.  Version: 2.2.0        (SPICE Toolkit N0057)

Summary for: mer2_crus_sim_id.bsp

Body: -255
      Start of Interval (ET)                  End of Interval (ET)
      --------------------------------    --------------------------------
      2003 JUL 09 00:15:00.000                2004 JAN 04 04:25:42.557

$ bspidmod -spki mer2_crus_sim_id.bsp -idi -255 -ido -254 -mod target -oflg
 The file mer2_crus_sim_id.bsp has been updated.

$ brief mer2_crus_sim_id.bsp
Brief.  Version: 2.2.0        (SPICE Toolkit N0057)

Summary for: mer2_crus_sim_id.bsp

Body: MER-2 (-254)
      Start of Interval (ET)                  End of Interval (ET)
      --------------------------------    --------------------------------
      2003 JUL 09 00:15:00.000                2004 JAN 04 04:25:42.557
```

DAFMOD

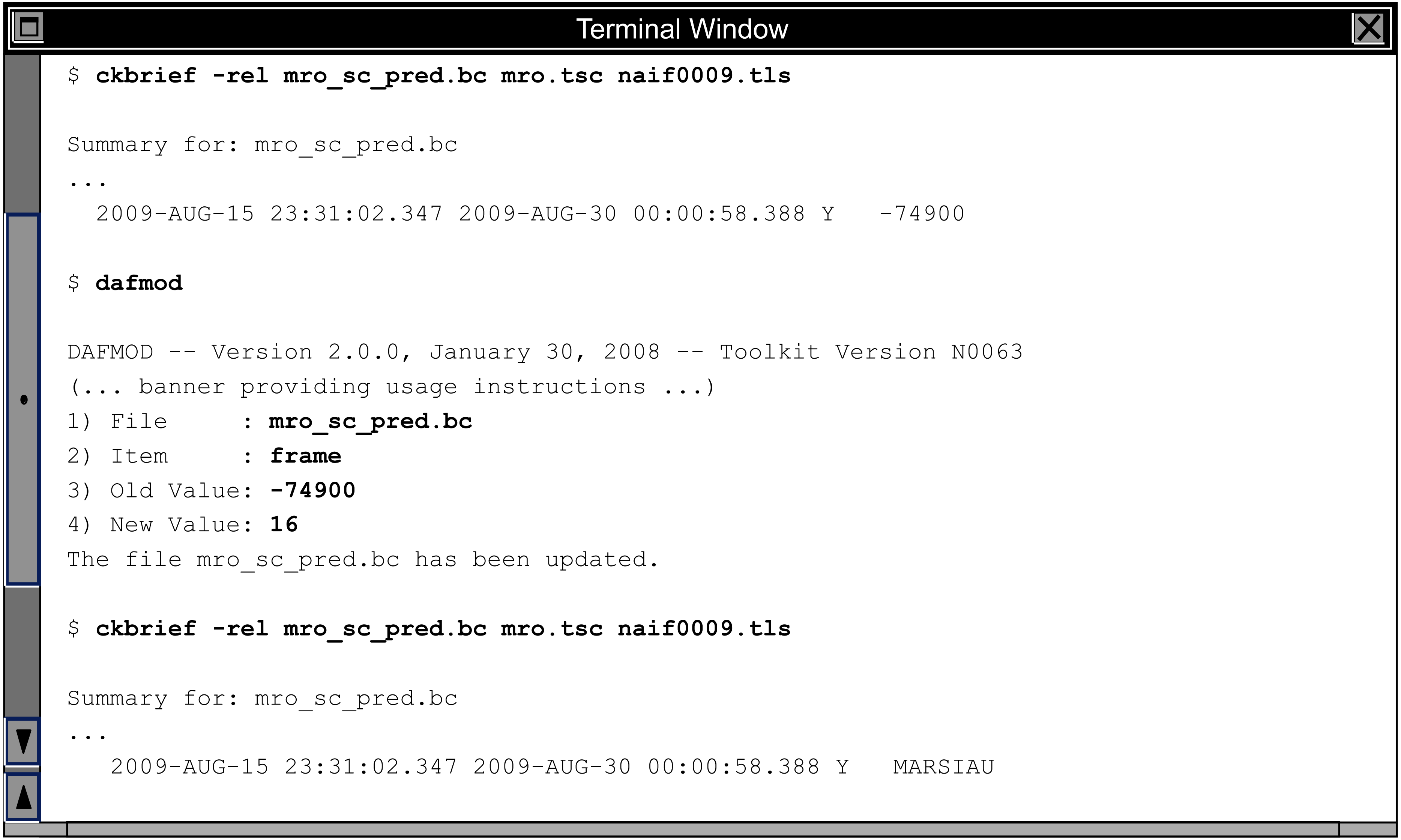
- *dafmod* is a program for altering selected segment attributes in a binary SPK, CK, or PCK file
  - in an SPK file it can alter the target, center, or reference frame ID
  - in a CK or binary PCK file it can alter the object or reference frame ID
- *dafmod* is an interactive program. When executed it prompts the user for
  - name of the file to be modified
  - “item” to be modified
    - the set of items depends on the kernel type
  - “old” item value
  - “new” item value
- *dafmod* puts into the comment area a warning note stating which items in which segments of the file were changed
- *dafmod* works only on files in native binary format
  - *bingo* may be used to convert a non-native binary kernel to native binary format

DAFMOD Example: SPK



```
$ brief mer2_crus_sim_id.bsp

Summary for: mer2_crus_sim_id.bsp

Body: -255
...

$ dafmod

DAFMOD -- Version 2.0.0, January 30, 2008 -- Toolkit Version N0063
(... banner providing usage instructions ...)
1) File     : mer2_crus_sim_id.bsp
2) Item     : target
3) Old Value: -255
4) New Value: -254
The file mer2_crus_sim_id.bsp has been updated.

$ brief mer2_crus_sim_id.bsp

Summary for: mer2_crus_sim_id.bsp

Body: MER-2 (-254)
```

DAFMOD Example: CK



```
$ ckbrief -rel mro_sc_pred.bc mro.tsc naif0009.tls

Summary for: mro_sc_pred.bc
...
  2009-AUG-15 23:31:02.347 2009-AUG-30 00:00:58.388 Y   -74900

$ dafmod

DAFMOD -- Version 2.0.0, January 30, 2008 -- Toolkit Version N0063
(... banner providing usage instructions ...)
1) File      : mro_sc_pred.bc
2) Item      : frame
3) Old Value: -74900
4) New Value: 16
The file mro_sc_pred.bc has been updated.

$ ckbrief -rel mro_sc_pred.bc mro.tsc naif0009.tls

Summary for: mro_sc_pred.bc
...
   2009-AUG-15 23:31:02.347 2009-AUG-30 00:00:58.388 Y   MARSIAU
```

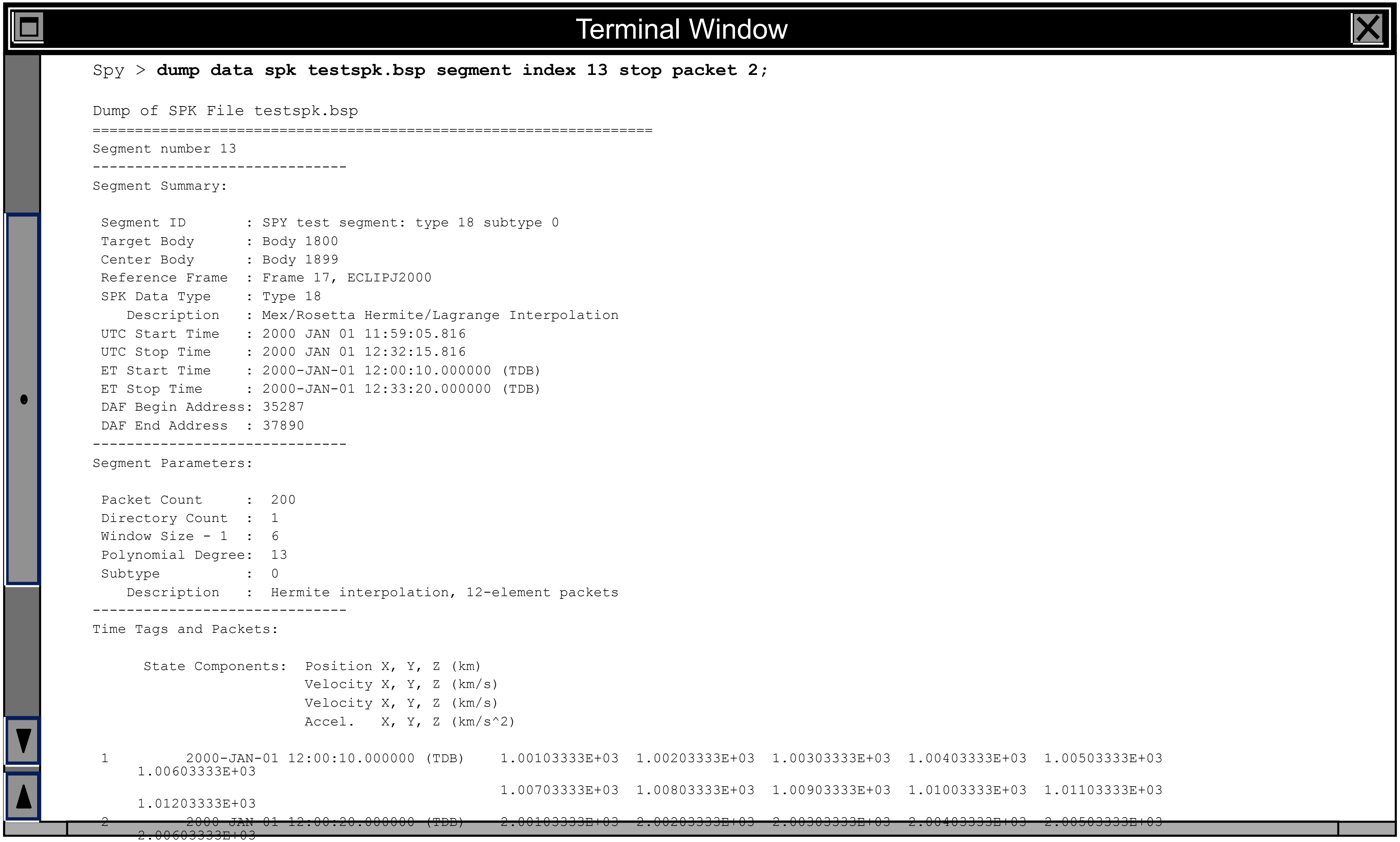
SPY

- *Spy* is a command-line utility for validating, inspecting, and analyzing SPK files
- Spy can:
  - Check SPK files
    - Validate SPK structure
    - Check sampled data for bounds violations
    - Locate invalid double precision numbers
  - Sample data from a set of loaded kernels
    - Sample position, distance, velocity, derived velocity, speed, acceleration, acceleration magnitude, osculating elements
  - Dump SPK file contents
    - Data
    - Summary information
    - Comment area
    - Bookkeeping information
  - Find some geometric events
    - Distance: find times when specified constraints on observer-target distance are met
    - Elevation: find times when specified constraints on elevation of target in specified frame are met

SPY: Selected Features

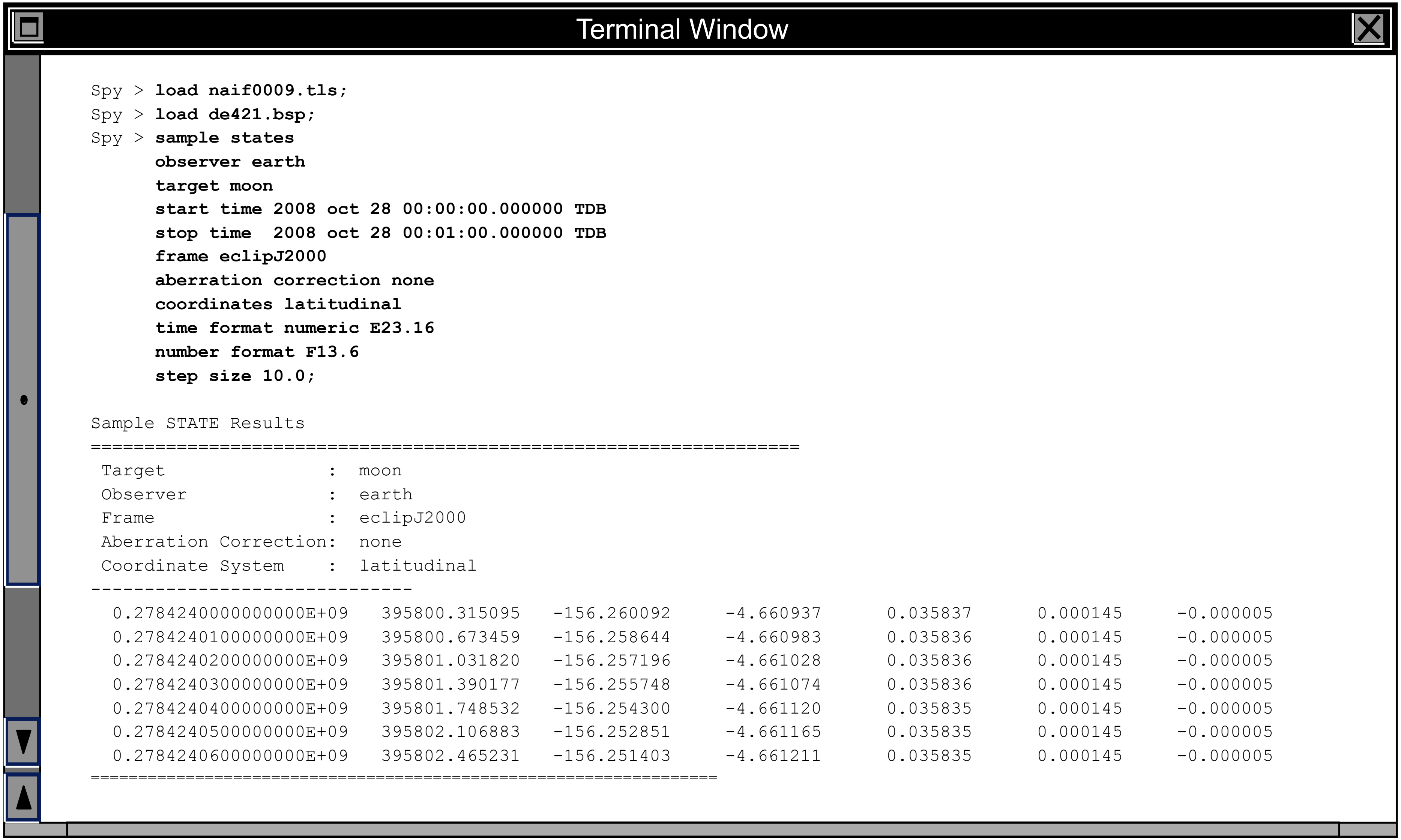
- Operating modes
  - Interactive, batch, shell command line
- Auxiliary files
  - Start-up file, command files, log file, save file
- Interactive command support
  - Command history: recall, repetition, and command editing; editor selection; command error detection; (limited) automatic command error correction
- User default support
  - Set, show, reset default values
- Input options
  - Define user symbols in commands
  - Embed prompts in commands
- Output options
  - Dump subsets of SPK data
  - Show epoch and packet deltas in data dumps
  - Set sample count or density
  - Set time and number formats
  - Set angular units
  - Set coordinate system for sampled data
  - Control error diagnostic verbosity
- Online help: command language summary

SPY Example: Dump SPK Data



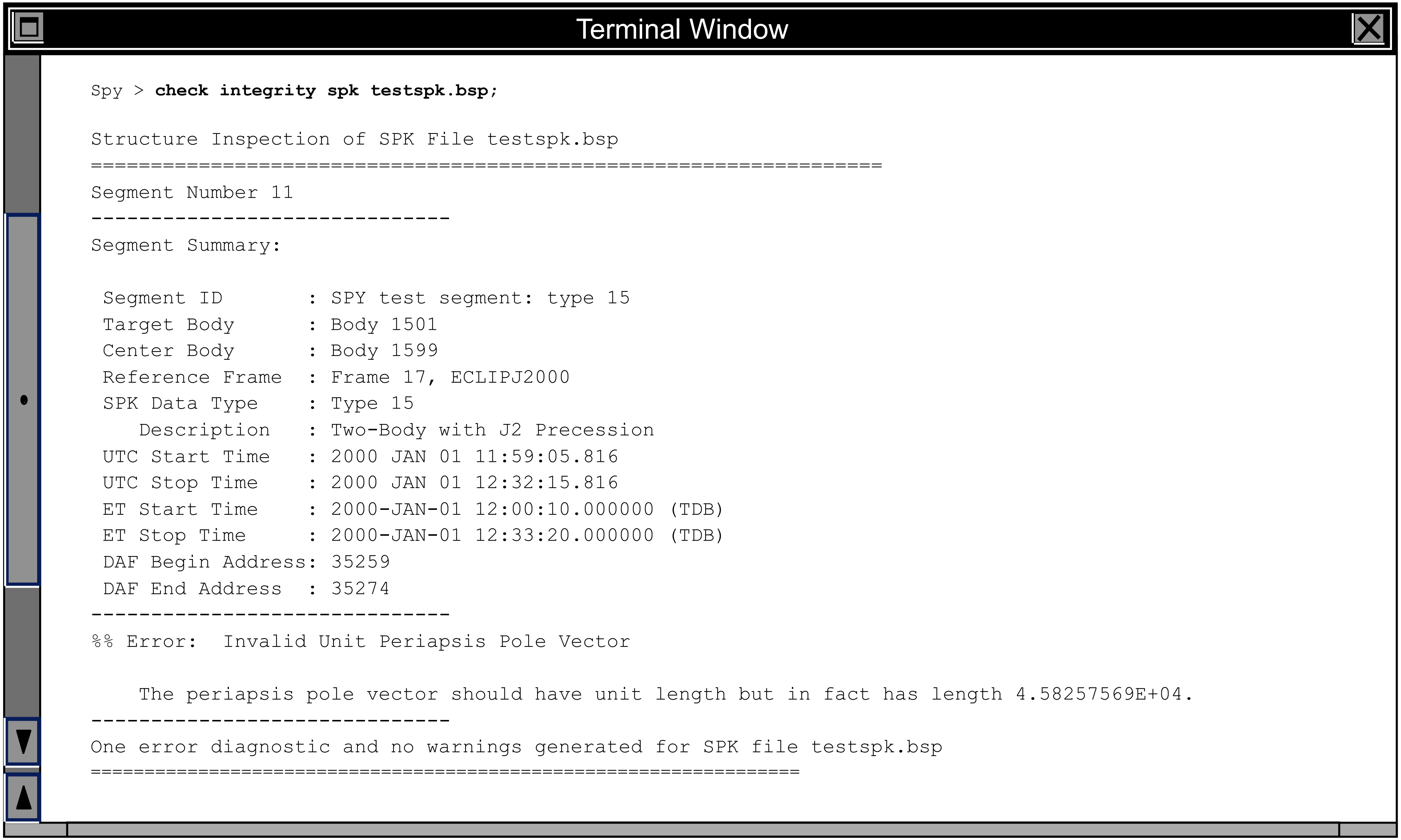
```
Spy > dump data spk testspk.bsp segment index 13 stop packet 2;

Dump of SPK File testspk.bsp
==================================================================
Segment number 13
------------------------------
Segment Summary:

 Segment ID         : SPY test segment: type 18 subtype 0
 Target Body        : Body 1800
 Center Body        : Body 1899
 Reference Frame    : Frame 17, ECLIPJ2000
 SPK Data Type      : Type 18
    Description     : Mex/Rosetta Hermite/Lagrange Interpolation
 UTC Start Time     : 2000 JAN 01 11:59:05.816
 UTC Stop Time      : 2000 JAN 01 12:32:15.816
 ET Start Time      : 2000-JAN-01 12:00:10.000000 (TDB)
 ET Stop Time       : 2000-JAN-01 12:33:20.000000 (TDB)
 DAF Begin Address: 35287
 DAF End Address    : 37890
------------------------------
Segment Parameters:

 Packet Count       :  200
 Directory Count    :  1
 Window Size - 1    :  6
 Polynomial Degree:  13
 Subtype            :  0
    Description     :  Hermite interpolation, 12-element packets
------------------------------
Time Tags and Packets:

      State Components:  Position X, Y, Z (km)
                         Velocity X, Y, Z (km/s)
                         Velocity X, Y, Z (km/s)
                         Accel.   X, Y, Z (km/s^2)

 1         2000-JAN-01 12:00:10.000000 (TDB)    1.00103333E+03  1.00203333E+03  1.00303333E+03  1.00403333E+03  1.00503333E+03
     1.00603333E+03
                                                1.00703333E+03  1.00803333E+03  1.00903333E+03  1.01003333E+03  1.01103333E+03
     1.01203333E+03
 2         2000-JAN-01 12:00:20.000000 (TDB)    2.00103333E+03  2.00203333E+03  2.00303333E+03  2.00403333E+03  2.00503333E+03
     2.00603333E+03
                                                2.00703333E+03  2.00803333E+03  2.00903333E+03  2.01003333E+03  2.01103333E+03
     2.01203333E+03
```

SPY Example: Sample State Vectors



```
Spy > load naif0009.tls;
Spy > load de421.bsp;
Spy > sample states
      observer earth
      target moon
      start time 2008 oct 28 00:00:00.000000 TDB
      stop time  2008 oct 28 00:01:00.000000 TDB
      frame eclipJ2000
      aberration correction none
      coordinates latitudinal
      time format numeric E23.16
      number format F13.6
      step size 10.0;

Sample STATE Results
==================================================================
 Target                :  moon
 Observer              :  earth
 Frame                 :  eclipJ2000
 Aberration Correction:  none
 Coordinate System     :  latitudinal
------------------------------
  0.2784240000000000E+09   395800.315095   -156.260092     -4.660937      0.035837      0.000145     -0.000005
  0.2784240100000000E+09   395800.673459   -156.258644     -4.660983      0.035836      0.000145     -0.000005
  0.2784240200000000E+09   395801.031820   -156.257196     -4.661028      0.035836      0.000145     -0.000005
  0.2784240300000000E+09   395801.390177   -156.255748     -4.661074      0.035836      0.000145     -0.000005
  0.2784240400000000E+09   395801.748532   -156.254300     -4.661120      0.035835      0.000145     -0.000005
  0.2784240500000000E+09   395802.106883   -156.252851     -4.661165      0.035835      0.000145     -0.000005
  0.2784240600000000E+09   395802.465231   -156.251403     -4.661211      0.035835      0.000145     -0.000005
===========================================================
```

SPY Example: Check SPK Integrity



```
Spy > check integrity spk testspk.bsp;

Structure Inspection of SPK File testspk.bsp
==================================================================
Segment Number 11
------------------------------
Segment Summary:

 Segment ID        : SPY test segment: type 15
 Target Body       : Body 1501
 Center Body       : Body 1599
 Reference Frame   : Frame 17, ECLIPJ2000
 SPK Data Type     : Type 15
    Description    : Two-Body with J2 Precession
 UTC Start Time    : 2000 JAN 01 11:59:05.816
 UTC Stop Time     : 2000 JAN 01 12:32:15.816
 ET Start Time     : 2000-JAN-01 12:00:10.000000 (TDB)
 ET Stop Time      : 2000-JAN-01 12:33:20.000000 (TDB)
 DAF Begin Address: 35259
 DAF End Address   : 35274
------------------------------
%% Error:  Invalid Unit Periapsis Pole Vector

    The periapsis pole vector should have unit length but in fact has length 4.58257569E+04.
------------------------------
One error diagnostic and no warnings generated for SPK file testspk.bsp
===========================================================
```

PREDICKT

- *prediCkt* is a program for making CK files from a set of orientation specification rules and schedules defining when these rules are to be applied
  - has a simple command line interface
  - requires orientation and schedule specification to be provided in a setup file that follows the SPICE text kernel syntax
  - requires all supporting kernels -- SPK, PCK, etc -- to be loaded using a meta kernel
  - for more details see the "Making a CK Tutorial"

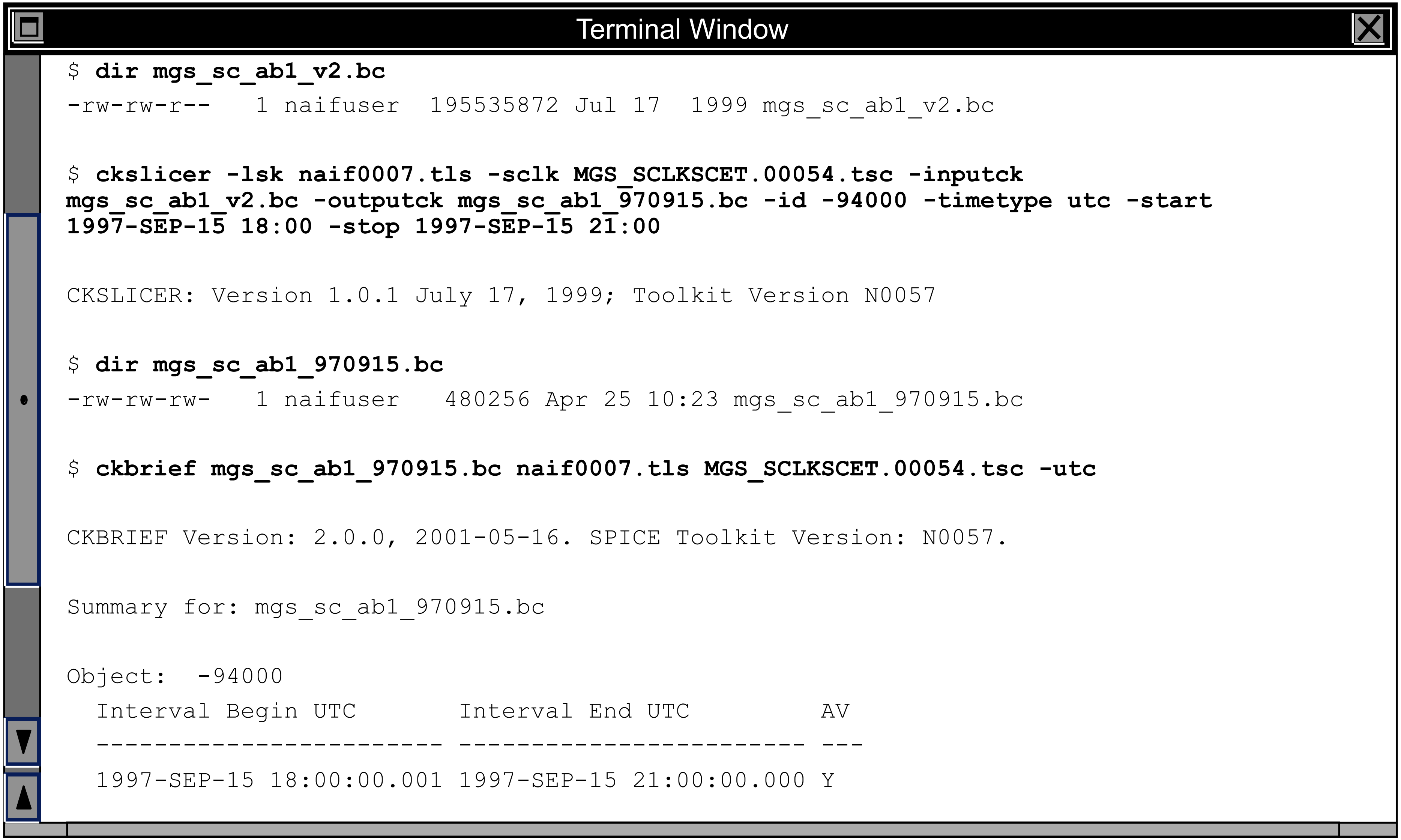
CKSLICER

- ***ckslicer*** **is a program for subsetting a CK file**
- ***ckslicer*** **is a command line program with the following usage**

```
        ckslicer -lsk <lsk_file_name>
                        -sclk <sclk_file_name(s)>
                        -inputck <ck_file_name>
                        -outputck <ck_file_name>
                        -id <naif_id>
                        -timetype <utc|sclk|ticks>
                        -start <start_time>
                        -stop <stop_time>
```

- ***ckslicer*** **is useful in the situation when only a portion of a CK covering a short interval of time is needed (for example when the whole CK is not needed and it takes up a lot of space) or to cut parts from a few CKs with the intent to merge them together (if reconstructed CKs from different sources have too much overlap to simply "cat" them together)**
- **A note stating which subset was extracted is put into the comment area of the output CK file**

CKSLICER Example



```
$ dir mgs_sc_ab1_v2.bc
-rw-rw-r--   1 naifuser  195535872 Jul 17  1999 mgs_sc_ab1_v2.bc

$ ckslicer -lsk naif0007.tls -sclk MGS_SCLKSCET.00054.tsc -inputck
mgs_sc_ab1_v2.bc -outputck mgs_sc_ab1_970915.bc -id -94000 -timetype utc -start
1997-SEP-15 18:00 -stop 1997-SEP-15 21:00

CKSLICER: Version 1.0.1 July 17, 1999; Toolkit Version N0057

$ dir mgs_sc_ab1_970915.bc
-rw-rw-rw-   1 naifuser   480256 Apr 25 10:23 mgs_sc_ab1_970915.bc

$ ckbrief mgs_sc_ab1_970915.bc naif0007.tls MGS_SCLKSCET.00054.tsc -utc

CKBRIEF Version: 2.0.0, 2001-05-16. SPICE Toolkit Version: N0057.

Summary for: mgs_sc_ab1_970915.bc

Object:  -94000
  Interval Begin UTC       Interval End UTC         AV
  ------------------------ ------------------------ ---
  1997-SEP-15 18:00:00.001 1997-SEP-15 21:00:00.000 Y
```

CKSPANIT

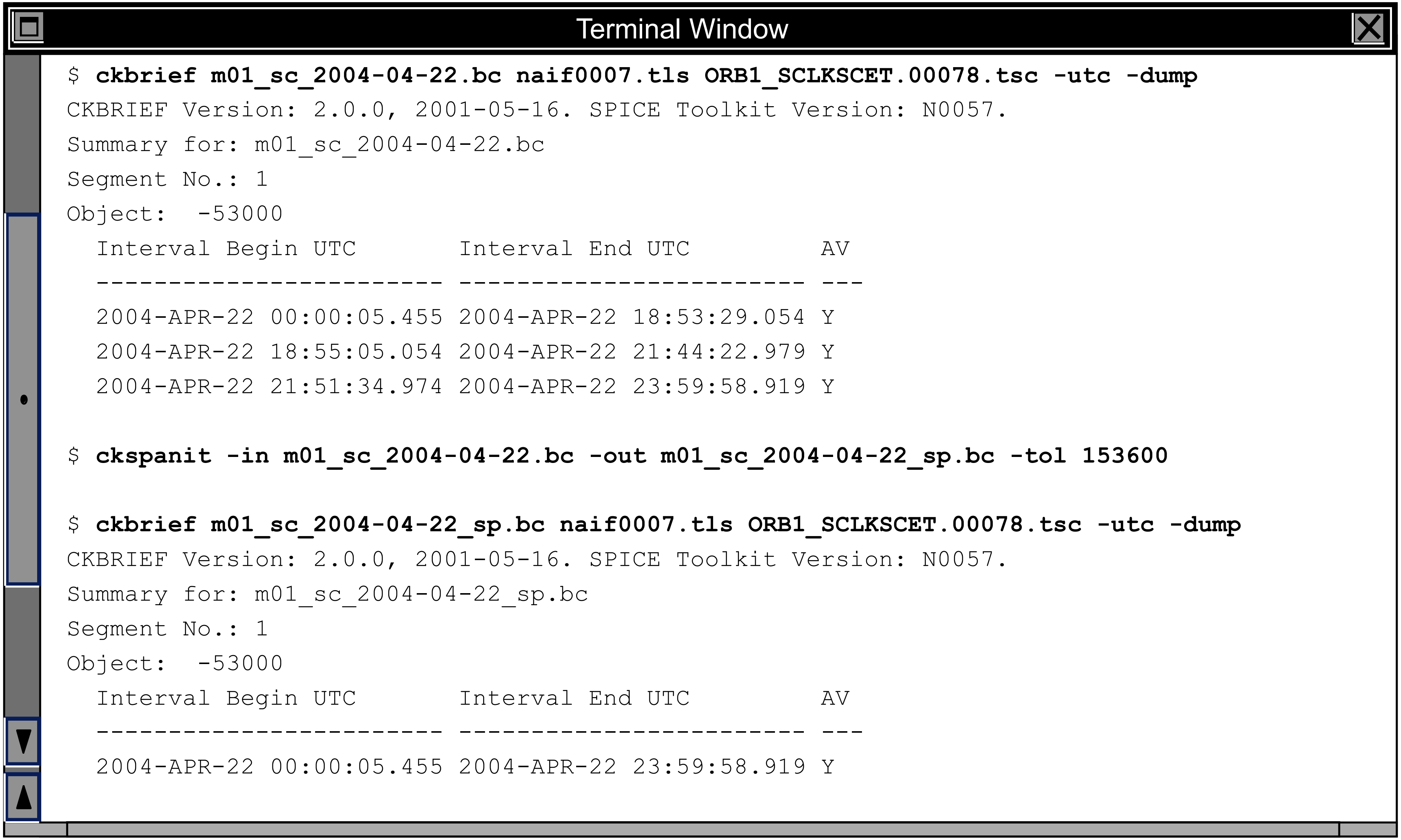
- *ckspanit* is a program for modifying interpolation interval information in type 3 CK segments
  - it can also convert a type 1 CK to a type 2 or 3 CK
- *ckspanit* is used when one is dealing with a type 3 CK containing many small gaps within segments. It allows you to alter the CK in such a way that SPICE will interpolate over those gaps
- *ckspanit* is a command line program with the following usage

```
ckspanit -in inp_ck -out out_ck -tol threshold [-frm fk]
```

  - "threshold" is the longest time interval over which interpolation is to be permitted in the output CK file
    - » must be specified in SCLK ticks
      - For example if 1 tick is 1/256 of a second and interpolation over 30 second intervals is needed, "threshold" must be set to 256*30=7680
  - "fk" is an optional FK file name, needed only if the base frame in the input CK is not one of the frames built into the Toolkit

CAUTION: before running *ckspanit*, make sure that interpolation over larger gaps is appropriate for the vehicle or structure you are dealing with. And don't forget to add appropriate comments to the newly created CK file.

CKSPANIT Example



```
$ ckbrief m01_sc_2004-04-22.bc naif0007.tls ORB1_SCLKSCET.00078.tsc -utc -dump
CKBRIEF Version: 2.0.0, 2001-05-16. SPICE Toolkit Version: N0057.
Summary for: m01_sc_2004-04-22.bc
Segment No.: 1
Object:  -53000
  Interval Begin UTC       Interval End UTC         AV
  ------------------------ ------------------------ ---
  2004-APR-22 00:00:05.455 2004-APR-22 18:53:29.054 Y
  2004-APR-22 18:55:05.054 2004-APR-22 21:44:22.979 Y
  2004-APR-22 21:51:34.974 2004-APR-22 23:59:58.919 Y

$ ckspanit -in m01_sc_2004-04-22.bc -out m01_sc_2004-04-22_sp.bc -tol 153600

$ ckbrief m01_sc_2004-04-22_sp.bc naif0007.tls ORB1_SCLKSCET.00078.tsc -utc -dump
CKBRIEF Version: 2.0.0, 2001-05-16. SPICE Toolkit Version: N0057.
Summary for: m01_sc_2004-04-22_sp.bc
Segment No.: 1
Object:  -53000
  Interval Begin UTC       Interval End UTC         AV
  ------------------------ ------------------------ ---
  2004-APR-22 00:00:05.455 2004-APR-22 23:59:58.919 Y
```

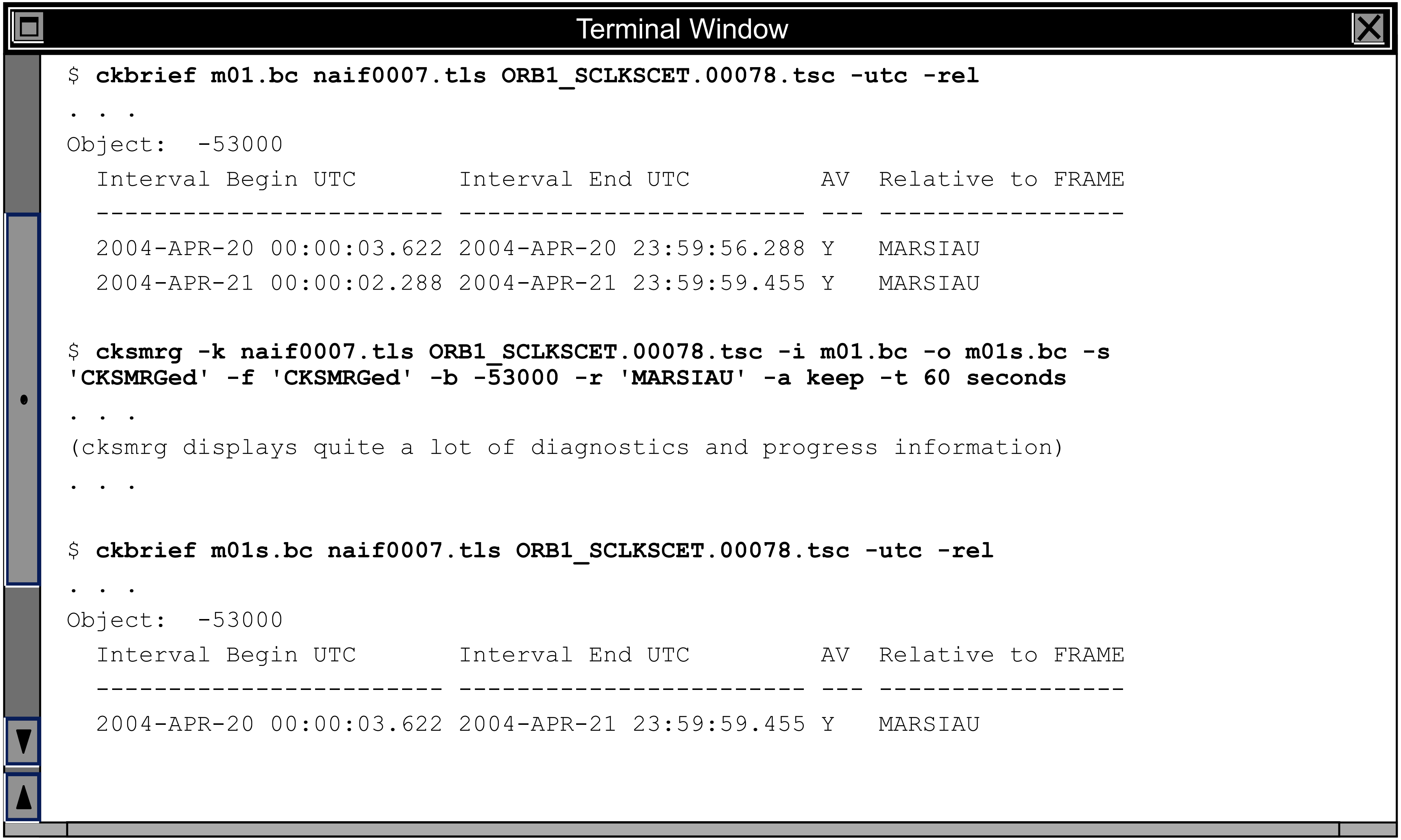
CKSMRG

- *cksmrg* is a program for merging data from two or more uniform CK segments (same ID, base frame and type) provided in a single CK file
- *cksmrg* is used for eliminating gaps between segments (that cannot be removed by *ckspanit*) and removing duplicate data points contained in different segments
- *cksmrg* is a command line program with the following usage

```
cksmrg -k|-kernels <meta kernel name|kernel file names>
       -i|-input <input ck file name>
       -o|-output <output ck file name>
       -s|-segid <output ck segment id string>
       -f|-fileid <output ck file id string>
       -b|-body <body id|name>
       -r|-reference <reference id|name>
       -a|-av <drop|keep|make|makeavrg>
       -t|-tolerance <tolerance (number units)>
       [-c|-correction <time delta|cor. table file>]
```

CAUTION: cksmrg should not be used to merge CKs from different sources, nor should it be used to merge overlapping predict CKs

CKSMRG Example

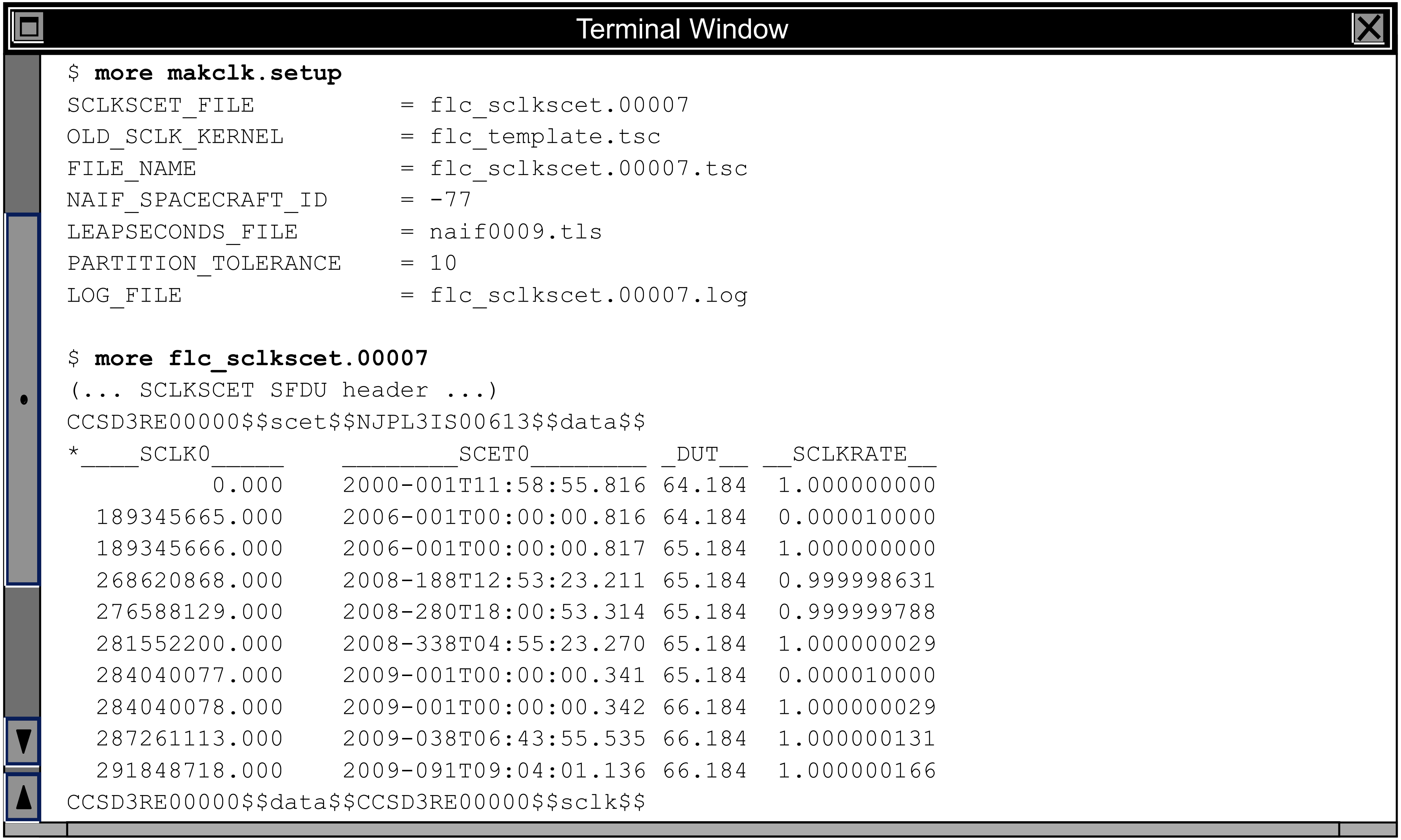


```
$ ckbrief m01.bc naif0007.tls ORB1_SCLKSCET.00078.tsc -utc -rel
. . .
Object:  -53000
  Interval Begin UTC        Interval End UTC          AV  Relative to FRAME
  ------------------------ ------------------------ --- -----------------
  2004-APR-20 00:00:03.622 2004-APR-20 23:59:56.288 Y   MARSIAU
  2004-APR-21 00:00:02.288 2004-APR-21 23:59:59.455 Y   MARSIAU

$ cksmrg -k naif0007.tls ORB1_SCLKSCET.00078.tsc -i m01.bc -o m01s.bc -s
'CKSMRGed' -f 'CKSMRGed' -b -53000 -r 'MARSIAU' -a keep -t 60 seconds
. . .
(cksmrg displays quite a lot of diagnostics and progress information)
. . .

$ ckbrief m01s.bc naif0007.tls ORB1_SCLKSCET.00078.tsc -utc -rel
. . .
Object:  -53000
  Interval Begin UTC        Interval End UTC          AV  Relative to FRAME
  ------------------------ ------------------------ --- -----------------
  2004-APR-20 00:00:03.622 2004-APR-21 23:59:59.455 Y   MARSIAU
```

MAKCLK

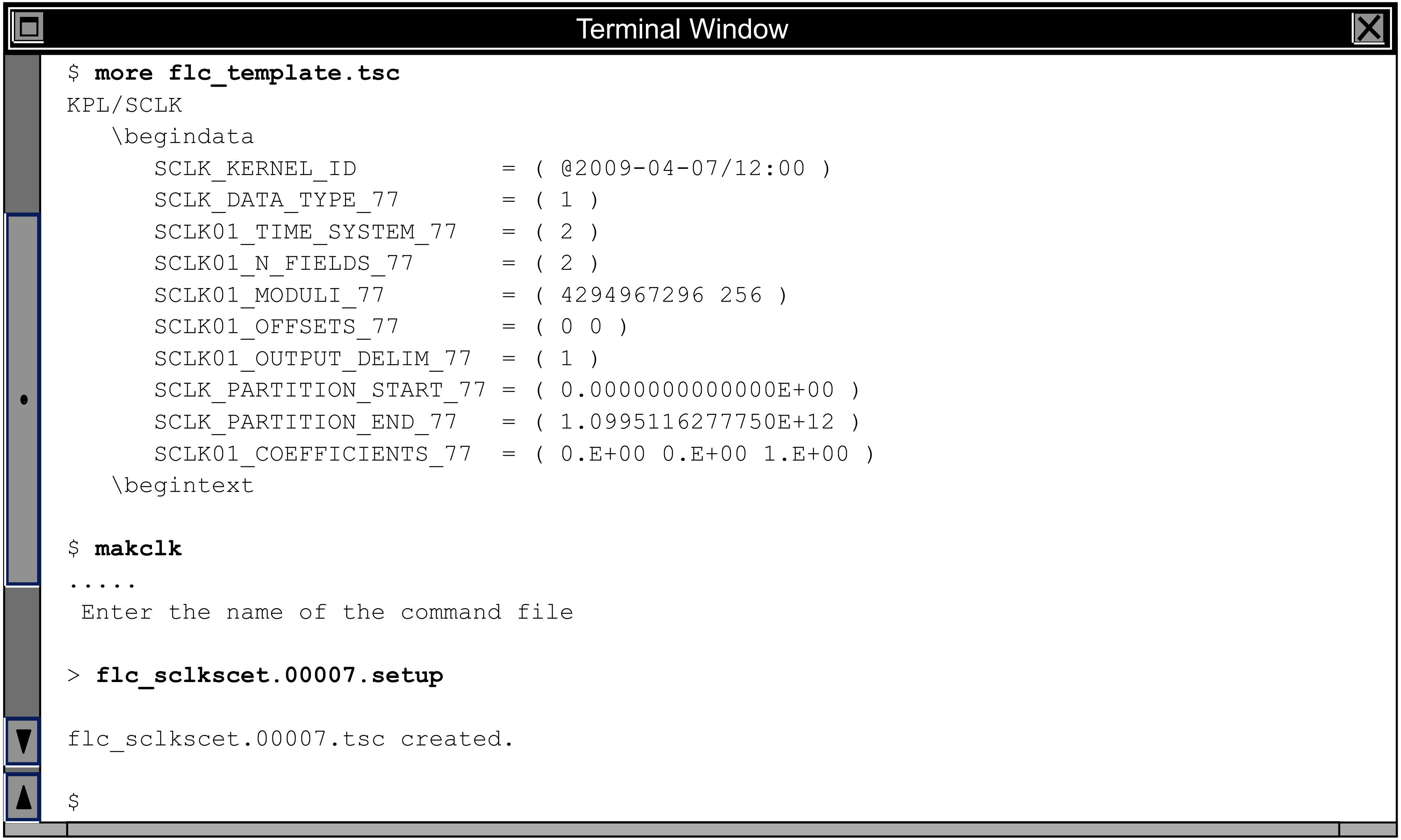
- *makclk* is a program for converting a SCLKSCET file to an SCLK kernel
  - SCLKSCET is a time correlation file used by most JPL missions
  - it is an ASCII text file providing piece-wise linear clock correlation function as an array of triplets consisting of the reference on-board time, the reference UTC time and the clock rate
  - NAIF found that in many cases it is much easier to write an application to first make a SCLKSCET file and then convert it to an SCLK kernel using *makclk* than to write an application to make an SCLK kernel from “scratch”
- *makclk* is an interactive program prompting for a single input - the name of the setup file
- The setup file uses KEYWORD=VALUE assignments to specify input files (SCLKSCET, template SCLK, and LSK), output files (SCLK kernel and log), and control parameters (spacecraft ID, partition tolerance, time filtering flag, and rate adjustment flag)
- The *makclk* User’s Guide provides detailed information about the setup file parameters and the SCLKSCET file format and contents.

MAKCLK Example



```
$ more makclk.setup
SCLKSCET_FILE          = flc_sclkscet.00007
OLD_SCLK_KERNEL        = flc_template.tsc
FILE_NAME              = flc_sclkscet.00007.tsc
NAIF_SPACECRAFT_ID     = -77
LEAPSECONDS_FILE       = naif0009.tls
PARTITION_TOLERANCE    = 10
LOG_FILE               = flc_sclkscet.00007.log

$ more flc_sclkscet.00007
(... SCLKSCET SFDU header ...)
CCSD3RE00000$$scet$$NJPL3IS00613$$data$$
*____SCLK0_____    ________SCET0________ _DUT__ __SCLKRATE__
          0.000    2000-001T11:58:55.816 64.184  1.000000000
  189345665.000    2006-001T00:00:00.816 64.184  0.000010000
  189345666.000    2006-001T00:00:00.817 65.184  1.000000000
  268620868.000    2008-188T12:53:23.211 65.184  0.999998631
  276588129.000    2008-280T18:00:53.314 65.184  0.999999788
  281552200.000    2008-338T04:55:23.270 65.184  1.000000029
  284040077.000    2009-001T00:00:00.341 65.184  0.000010000
  284040078.000    2009-001T00:00:00.342 66.184  1.000000029
  287261113.000    2009-038T06:43:55.535 66.184  1.000000131
  291848718.000    2009-091T09:04:01.136 66.184  1.000000166
CCSD3RE00000$$data$$CCSD3RE00000$$sclk$$
```

MAKCLK Example (continued)



```
$ more flc_template.tsc
KPL/SCLK
   \begindata
      SCLK_KERNEL_ID            = ( @2009-04-07/12:00 )
      SCLK_DATA_TYPE_77         = ( 1 )
      SCLK01_TIME_SYSTEM_77     = ( 2 )
      SCLK01_N_FIELDS_77        = ( 2 )
      SCLK01_MODULI_77          = ( 4294967296 256 )
      SCLK01_OFFSETS_77         = ( 0 0 )
      SCLK01_OUTPUT_DELIM_77    = ( 1 )
      SCLK_PARTITION_START_77 = ( 0.0000000000000E+00 )
      SCLK_PARTITION_END_77     = ( 1.0995116277750E+12 )
      SCLK01_COEFFICIENTS_77    = ( 0.E+00 0.E+00 1.E+00 )
   \begintext

$ makclk
.....
 Enter the name of the command file

> flc_sclkscet.00007.setup

flc_sclkscet.00007.tsc created.

$
```

ORBNUM

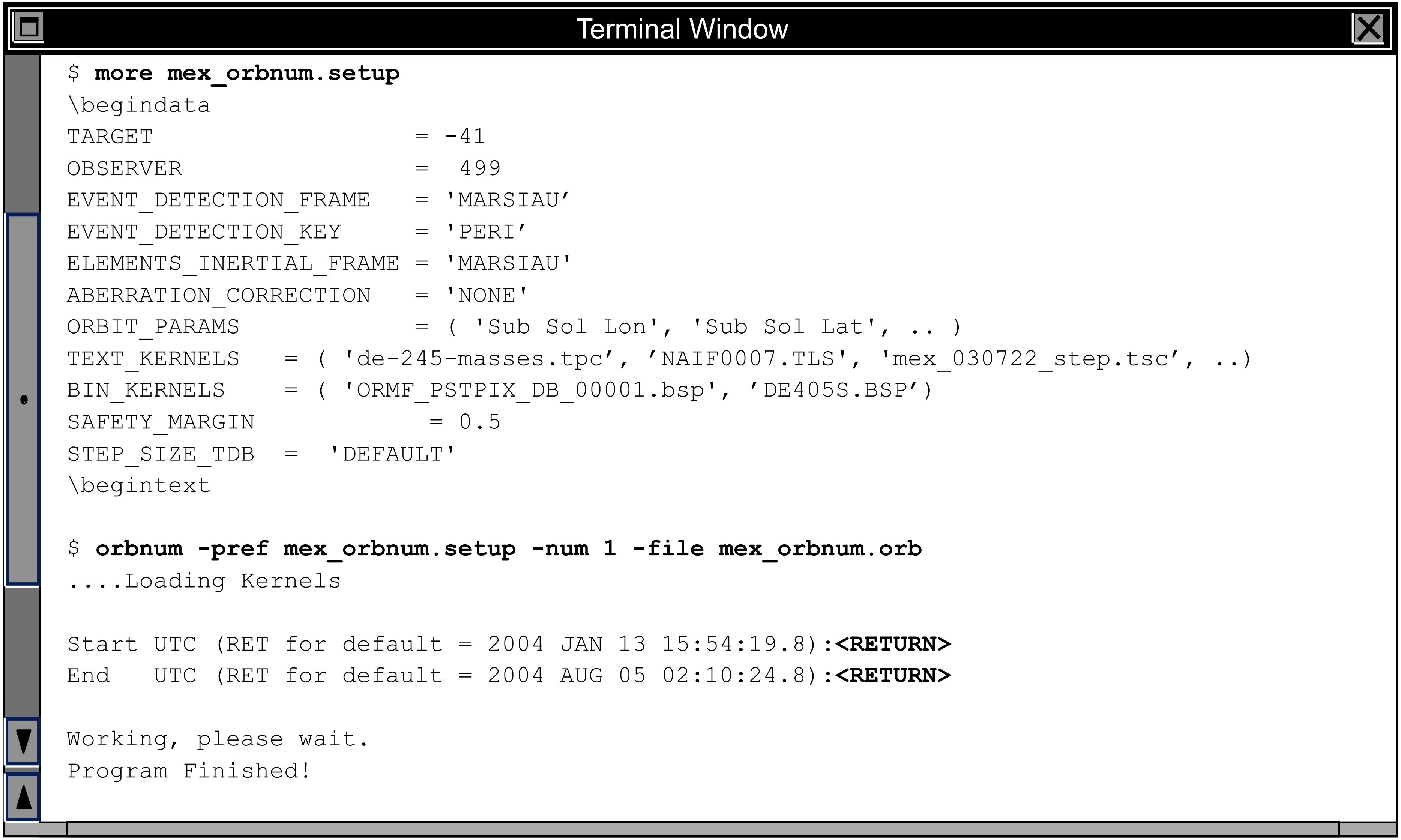
- *orbnum* is a program for generating a SPICE orbit number file containing orbit start/stop times and orbit numbers along with some additional derived quantities (orbital elements and coordinates of sub-spacecraft and sub-solar points)
  - The orbit number increment can be specified as occurring at one of these events: periapsis or apoapsis, ascending or descending equatorial node crossing, min or max value for the s/c position’s Z-coordinate, or min or max value of the s/c latitude
- *orbnum* is a command line program with the following usage

```
orbnum -pref pref_file -num init_orbit -file orbnum_file -d -v -audit -tdb -verbose
```

optional

  - “pref_file” is a preferences file using text kernel syntax, specifying setup parameters along with the kernels containing data to be used to search for orbit start and stop events -- spacecraft trajectory SPKs, center body PCK, spacecraft SCLK, etc.
  - “init_orbit” is the number to be assigned to the first orbit determined using the kernels provided; subsequent orbits are assigned by incrementing “init_orbit” by 1
  - “orbnum_file” is the name of the orbit number file to be created
- **An orbnum file is not considered a SPICE kernel**
  - It’s just a convenient, derived product that NAIF offers to make for orbital missions that wish to have it

ORBNUM Example



```
$ more mex_orbnum.setup
\begindata
TARGET                  = -41
OBSERVER                =  499
EVENT_DETECTION_FRAME   = 'MARSIAU’
EVENT_DETECTION_KEY     = 'PERI’
ELEMENTS_INERTIAL_FRAME = 'MARSIAU'
ABERRATION_CORRECTION   = 'NONE'
ORBIT_PARAMS            = ( 'Sub Sol Lon', 'Sub Sol Lat', .. )
TEXT_KERNELS    = ( 'de-245-masses.tpc’, ’NAIF0007.TLS', 'mex_030722_step.tsc’, ..)
BIN_KERNELS     = ( 'ORMF_PSTPIX_DB_00001.bsp', ’DE405S.BSP’)
SAFETY_MARGIN             = 0.5
STEP_SIZE_TDB   =  'DEFAULT'
\begintext

$ orbnum -pref mex_orbnum.setup -num 1 -file mex_orbnum.orb
....Loading Kernels

Start UTC (RET for default = 2004 JAN 13 15:54:19.8):<RETURN>
End   UTC (RET for default = 2004 AUG 05 02:10:24.8):<RETURN>

Working, please wait.
Program Finished!
```

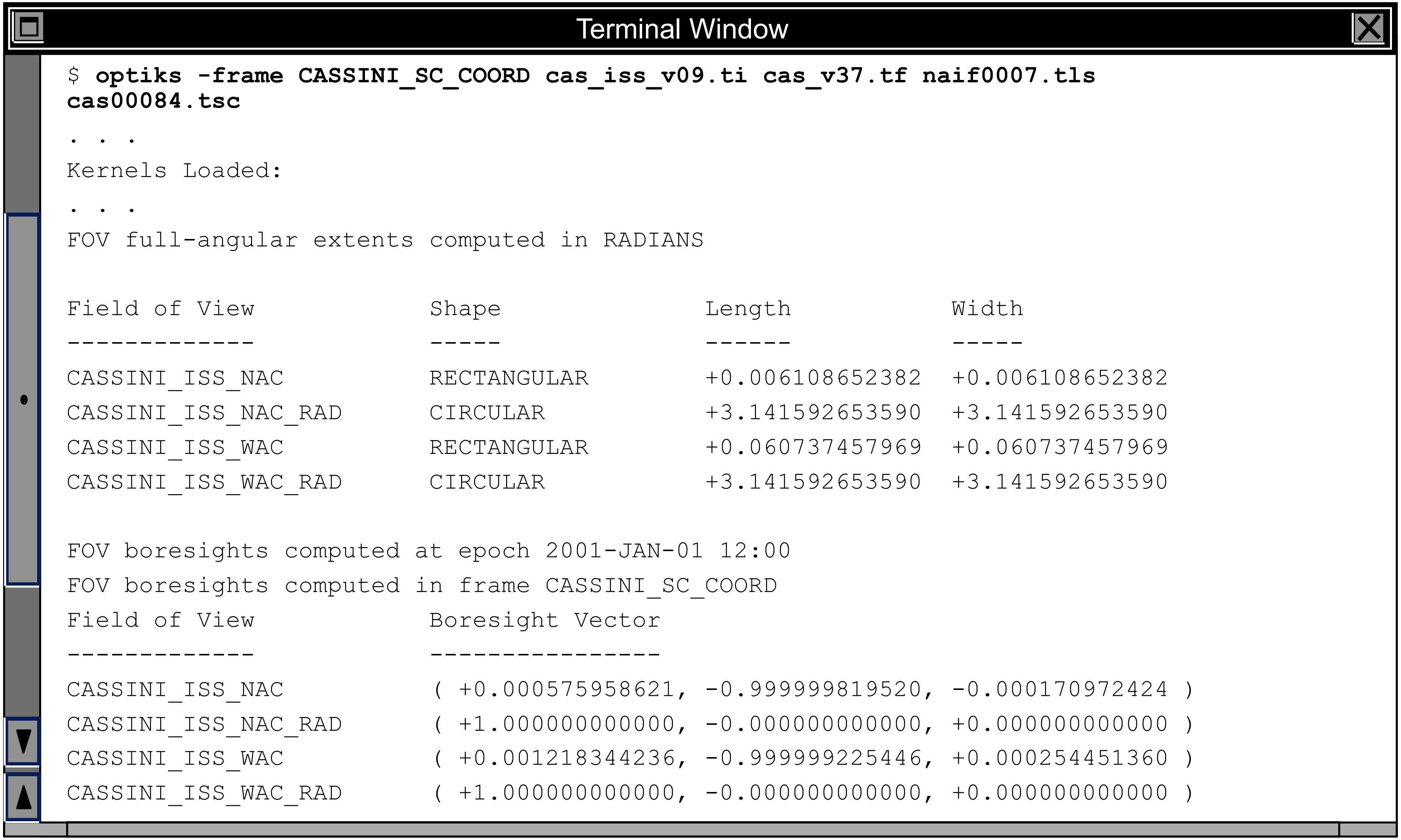
OPTIKS

- *optiks* is a utility program that generates information about instrument fields of view (FOV) from parameters present in IK and FK files
  - FOVs must be defined using the keywords required by the GETFOV routine
- *optiks* is a command line program used in one of two ways

```
optiks [options]... kernel ...
optiks [options]... meta-kernel ...
```

- *optiks* uses a set of SPICE kernels specified on the command line; one or more of these kernels may be a meta-kernel
- The output data are organized in three tables
  - The first table lists the angular extents (size) of circular, elliptical, and rectangular FOVs. Using command line options “-units” and “-half” the user can select the unit of measure for the angular measurements, and whether half or full FOV angular extents are listed.
  - The second table contains FOV boresights in a user specified frame at a particular epoch, specified using the “-epoch” option
  - The third shows FOV boundary vectors and boresights as returned from the GETFOV API, or unitized and rotated into a user-specified frame at a particular epoch

OPTIKS Example



```
$ optiks -frame CASSINI_SC_COORD cas_iss_v09.ti cas_v37.tf naif0007.tls
cas00084.tsc
. . .
Kernels Loaded:
. . .
FOV full-angular extents computed in RADIANS

Field of View            Shape              Length            Width
-------------            -----              ------            -----
CASSINI_ISS_NAC          RECTANGULAR        +0.006108652382   +0.006108652382
CASSINI_ISS_NAC_RAD      CIRCULAR           +3.141592653590   +3.141592653590
CASSINI_ISS_WAC          RECTANGULAR        +0.060737457969   +0.060737457969
CASSINI_ISS_WAC_RAD      CIRCULAR           +3.141592653590   +3.141592653590

FOV boresights computed at epoch 2001-JAN-01 12:00
FOV boresights computed in frame CASSINI_SC_COORD
Field of View            Boresight Vector
-------------            ----------------
CASSINI_ISS_NAC          ( +0.000575958621, -0.999999819520, -0.000170972424 )
CASSINI_ISS_NAC_RAD      ( +1.000000000000, -0.000000000000, +0.000000000000 )
CASSINI_ISS_WAC          ( +0.001218344236, -0.999999225446, +0.000254451360 )
CASSINI_ISS_WAC_RAD      ( +1.000000000000, -0.000000000000, +0.000000000000 )
```

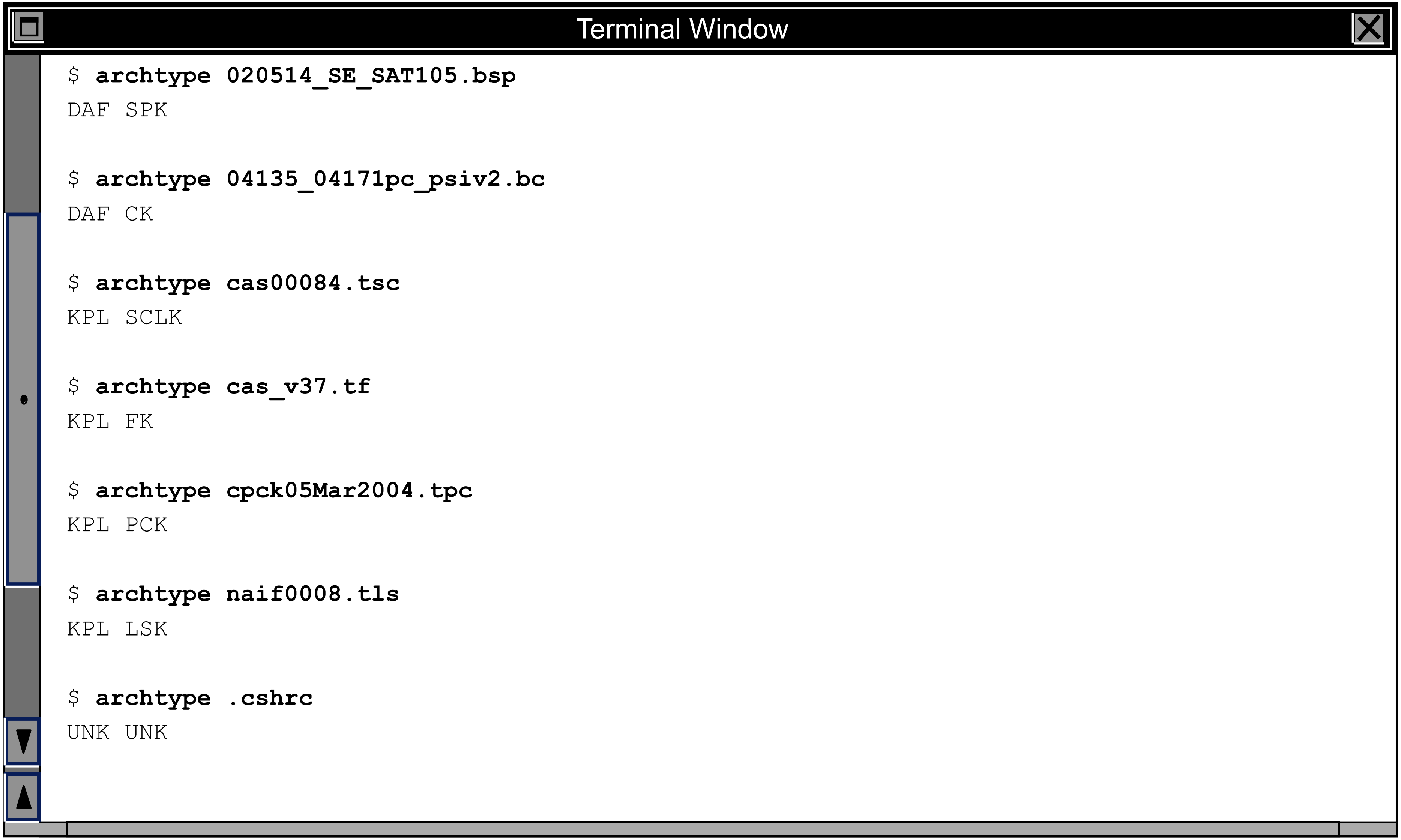
ARCHTYPE

- ***archtype*** **is a program that displays the file architecture and type of a SPICE kernel; it is useful for scripting applications**
  - To identify the architecture and type *archtype* uses the same mechanism as the FURNSH routine
- ***archtype*** **has a simple command line interface and requires only one argument -- the name of a kernel file:**

```
archtype kernel_name
```

- ***archtype*** **prints architecture and type to standard output as two space delimited acronyms**
  - Architecture can be:
    - » 'DAF' or 'DAS' for binary kernels
    - » 'KPL' for text kernels
  - Type can be 'SPK', 'PCK', 'IK', 'CK', 'EK', 'LSK', 'SCLK', 'FK', 'MK'
- **If architecture and/or type cannot be determined, the program displays 'UNK'**
- **In order for text kernels to be recognized, the first few characters of the file must contain 'KPL/<type>' (i.e. 'KPL/IK', 'KPL/FK', etc.)**

ARCHTYPE Examples



```
$ archtype 020514_SE_SAT105.bsp
DAF SPK

$ archtype 04135_04171pc_psiv2.bc
DAF CK

$ archtype cas00084.tsc
KPL SCLK

$ archtype cas_v37.tf
KPL FK

$ archtype cpck05Mar2004.tpc
KPL PCK

$ archtype naif0008.tls
KPL LSK

$ archtype .cshrc
UNK UNK
```

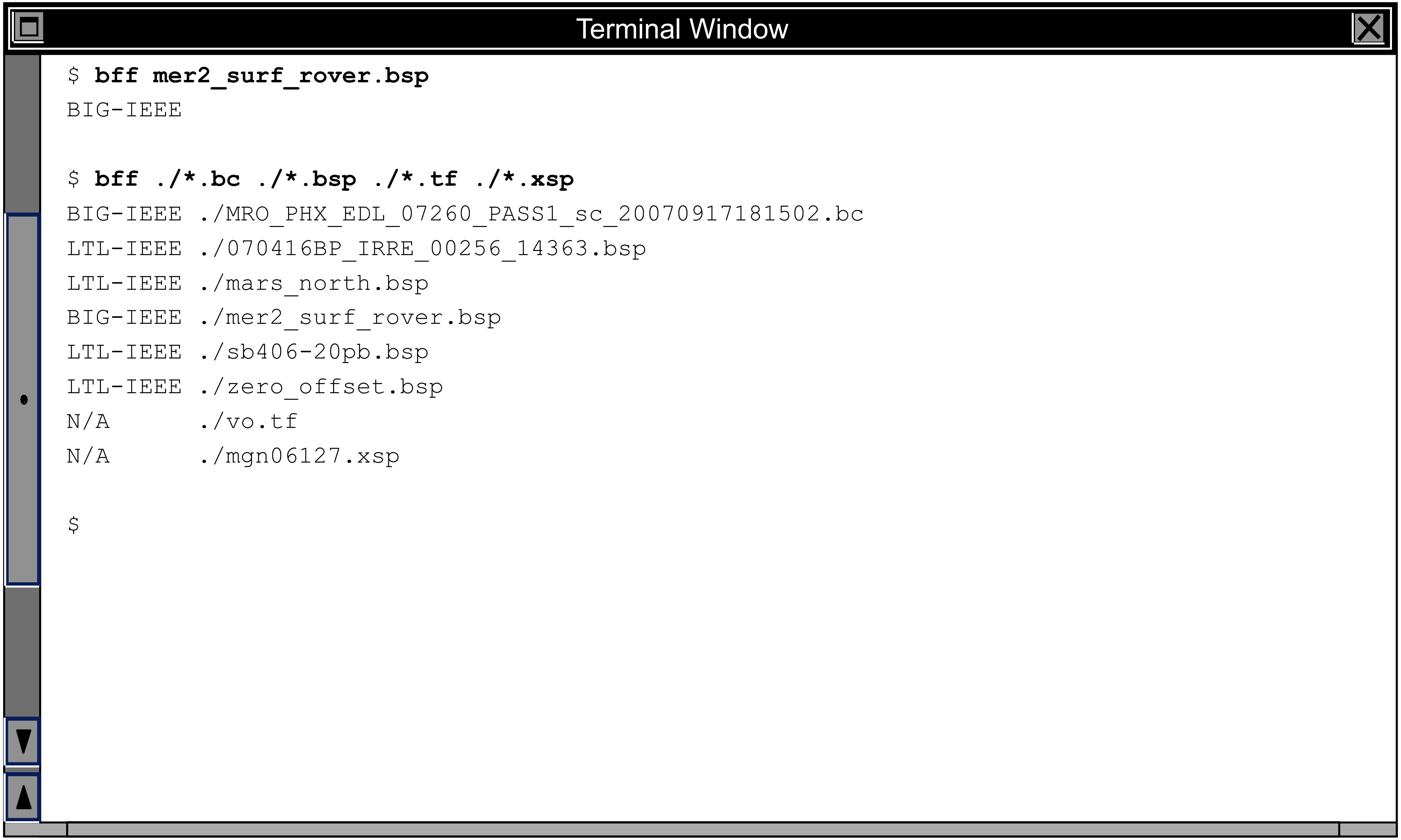
BFF

- *bff* is a program that displays the binary file format of one or a few SPICE kernels
- *bff* has a simple command line interface requiring kernel names to be listed on the command line:

```
bff kernel_name [kernel_name ...]
```

- *bff* prints the binary file format string (‘BIG-IEEE’ or ‘LTL-IEEE’) to standard output
  - when run on a single kernel, it prints only the format string
  - when run on more than one kernel, it prints the format string followed by the file name on a separate line for each file
- If an input file is not a binary kernel, the program displays the format string ‘N/A’
- If the binary file format cannot be determined (for DAS files produced by applications linked to SPICE Toolkit N0051, April 2000 or earlier), the program displays the format string ‘UNK’

BFF Examples



```
$ bff mer2_surf_rover.bsp
BIG-IEEE

$ bff ./*.bc ./*.bsp ./*.tf ./*.xsp
BIG-IEEE ./MRO_PHX_EDL_07260_PASS1_sc_20070917181502.bc
LTL-IEEE ./070416BP_IRRE_00256_14363.bsp
LTL-IEEE ./mars_north.bsp
BIG-IEEE ./mer2_surf_rover.bsp
LTL-IEEE ./sb406-20pb.bsp
LTL-IEEE ./zero_offset.bsp
N/A      ./vo.tf
N/A      ./mgn06127.xsp

$
```

BINGO

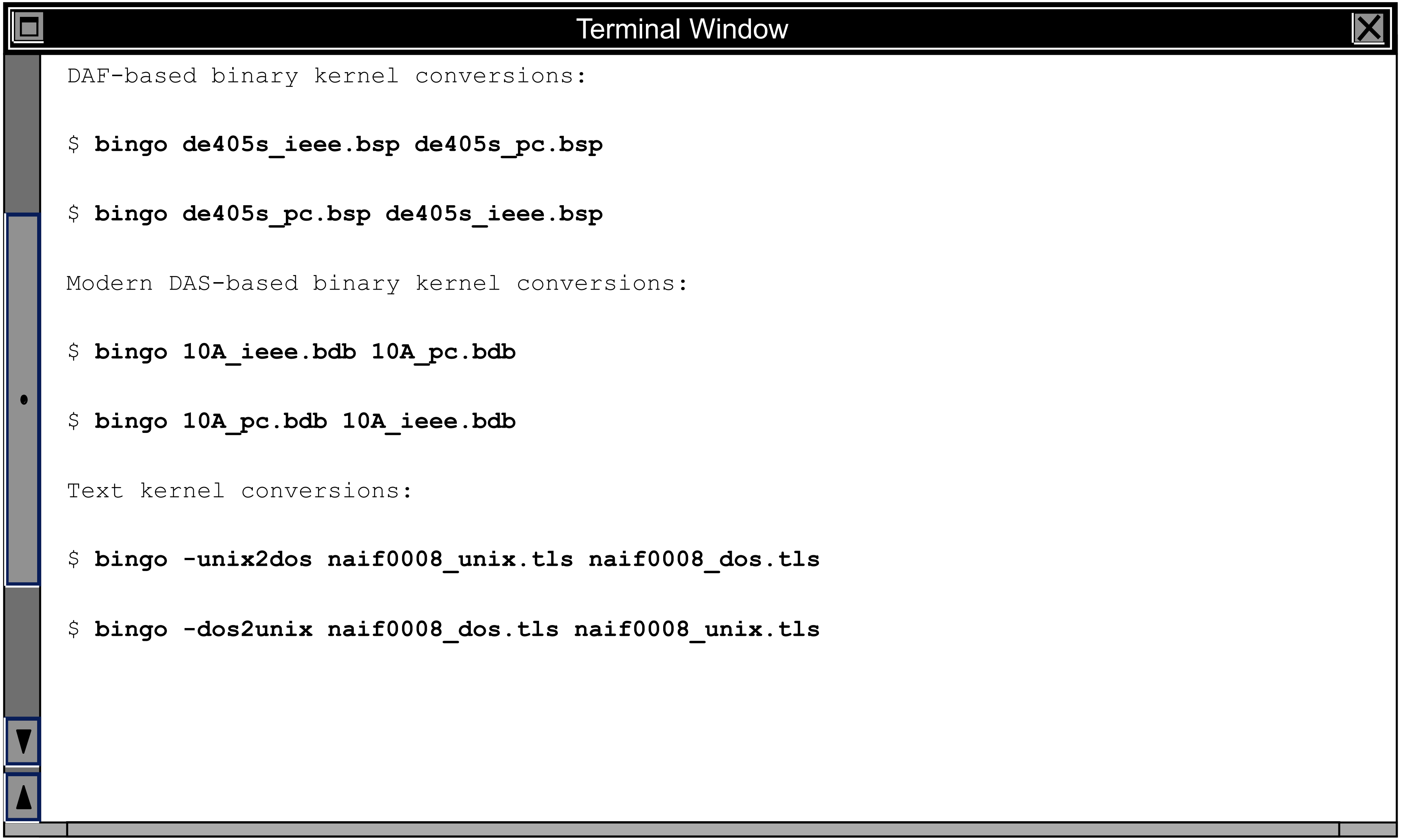
- *bingo* is a program that converts:
  - binary SPICE kernels between IEEE big endian and little endian formats
  - text format SPICE kernels between DOS and UNIX text formats
- *bingo* has a simple command line interface:

```
bingo [option] input_kernel output_kernel
```

  - "option" is a flag specifying the conversion direction: '-ieee2pc' or '-pc2ieee' for binary kernels and '-unix2dos' or '-dos2unix' for text format kernels
  - "input_kernel" is the input kernel file name
  - "output_kernel" is the output kernel file name. If the output file exists, the program overwrites it.
- The conversion direction flag does not need to be specified for DAF-based binary file conversions (SPK, CK, binary PCK) and post-N0051 DAS-based binary file conversions (EK, DBK, DSK)
  - The program automatically determines the input file format and performs conversion to the other format
- The conversion direction flag must be specified for pre-N0051 DAS-based binary file conversions, and for text file conversions

BINGO Examples

Navigation and Ancillary Information Facility



```
DAF-based binary kernel conversions:

$ bingo de405s_ieee.bsp de405s_pc.bsp

$ bingo de405s_pc.bsp de405s_ieee.bsp

Modern DAS-based binary kernel conversions:

$ bingo 10A_ieee.bdb 10A_pc.bdb

$ bingo 10A_pc.bdb 10A_ieee.bdb

Text kernel conversions:

$ bingo -unix2dos naif0008_unix.tls naif0008_dos.tls

$ bingo -dos2unix naif0008_dos.tls naif0008_unix.tls
```