Chapter 25.
Functions Ma thru Rz

Topic: Ignore

make_Scm_Entity

Function: Scheme Interface, Entity

Action: Creates a Scheme entity from a C++ ENTITY.

Prototype:
```
ScmObject make_Scm_Entity (
    ENTITY* ent             // entity
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kern_scm/ent_typ.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”scheme/elk/object.h”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: kern_scm

Filename: kern/kern_scm/ent_typ.hxx

Effect: Read–only

make_surface

Function: Construction Geometry, Extending ACIS

Action: Creates a surface for the given surface constant.

Prototype:
```
SURFACE* make_surface (
    surface const& this_surface // surface
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kerndata/geom/cnstruct.hxx″
#include ″kernel/kerndata/geom/surface.hxx″
#include ″kernel/kerngeom/surface/surdef.hxx″
```

Description: Used by the CURVE_constructor class.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kerndata/geom/cnstruct.hxx

Effect: Read–only

proj_pt_to_line

Function: Construction Geometry, Intersectors, Modifying Models

Action: Projects a SPAposition onto a line.

Prototype:
```
SPAposition proj_pt_to_line (
    const SPAposition& pt,      // position to
                                // project
    const SPAposition& line_pt, // position on line
    const SPAunit_vector& line_dir// direction of
                                // line
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/geomhusk/geom_utl.hxx″
#include ″baseutil/vector/position.hxx″
#include ″baseutil/vector/unitvec.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/geomhusk/geom_utl.hxx

Effect: Read–only

proj_pt_to_plane

Function: Construction Geometry, Intersectors, Modifying Models

Action: Projects a SPAposition onto a plane.

Prototype:
```
SPAposition proj_pt_to_plane (
    const SPAposition& pt,  // position to project
    const SPAposition& c,   // position on plane
    const SPAunit_vector& n // plane normal
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/geomhusk/geom_utl.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/unitvec.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/geomhusk/geom_utl.hxx

Effect: Read–only

read_array

Function: SAT Save and Restore

Action: Reads an of array indices.

Prototype:
```
ENTITY* read_array (
    ENTITY* array[],        // array of entities
    int i                   // number of entities
    );

ENTITY* read_array (
    ENTITY* array[],        // array of entities
    const void* ptr         // pointer to restore
                            // routine
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kerndata/savres/savres_small.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file. It returns an array of indices or NULL for negative index.

```
if (i < 0)
    return NULL
else
    return array[i]                    Array of indices.
```

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kerndata/savres/savres_small.hxx

Effect: Read–only

read_char

Function: SAT Save and Restore

Action: Reads a character written with C printf format “%c”.

Prototype:
```
int read_char ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernutil/fileio/fileio.hxx”
```

Description: This routine is used as part of restore from a SAT or SAB file. ActiveFile is a FileInterface object and does most of the actual work.

```
return ActiveFile ? ActiveFile–>read_char() : EOF;
                                    Call the appropriate SatFile or
                                    SabFile method
```

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_data

Function: SAT Save and Restore

Action: Reads a TaggedData item from an unknown ENTITY type.

Prototype: `TaggedData* read_data ();`

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kernutil/fileio/fileio.hxx″
#include ″kernel/kernutil/fileio/tagdata.hxx″
```

Description: This routine is used as part of restore from a SAT or SAB file. ActiveFile is a FileInterface object and does most of the actual work. Reads a TaggedData item from an unknown ENTITY type. This procedure returns a new object which is allocated on the heap. It is the callers responsibility to free it when it is done with it. Normally, the object will be appended to a TaggedDataList, and the list will assume responsibility for deleting it.

return ActiveFile ? ActiveFile–>read_data() : NULL;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_enum

Function: SAT Save and Restore

Action: Reads an enumeration table.

Prototype:
```
int read_enum (
    enum_table const& tbl   // enumeration table
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kernutil/fileio/fileio.hxx″
#include ″baseutil/mmgr/enum_tbl.hxx″
```

Description: Read an enumeration table. The <identifier> specifies which enumeration is active and its valid values. The <identifier> is not written to the file. A valid value only is written to the file. This is a character string or a long value from the enumeration <identifier> written with C printf format “%s”. For compatibility with older files, accept the integer value, even for interfaces which write the corresponding string. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_enum(tb1) : 0;

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_float

Function: SAT Save and Restore

Action: Reads a float written with C printf format “%g ”.

Prototype:
```
float read_float ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/meshhusk/mesh/node.hxx”
```

Description: This routine is used as part of restore from a SAT or SAB file. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_float() : 0;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/meshhusk/mesh/node.hxx

Effect: Read–only

read_header

Function: SAT Save and Restore

Action: Reads a header.

Prototype:
```
logical read_header (
    int& i1,                // release level
    int& i2,                // number of data records
    int& i3,                // number of entities
    int& i4                 // history
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
#include "baseutil/logical.h"
```

Description: Reads a header. The first record of the ACIS save file is a header, such as: 200 0 1 0

First Integer: An encoded version number. In the example, this is "200". This value is 100 times the major version plus the minor version (e.g., 107 for ACIS version 1.7). For point releases, the final value is truncated. Part save data for the .sat files is not affected by a point release (e.g., 105 for ACIS version 1.5.2).

Second Integer: The total number of saved data records, or zero. If zero, then there needs to be an end mark.

Third Integer: A count of the number of entities in the original entity list saved to the part file.

Fourth Integer: The least significant bit of this number is used to indicate whether or not history has been saved in this save file.

ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_header(i1, i2, i3, i4) : FALSE;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_id

Function: SAT Save and Restore

Action: Reads an identifier.

Prototype:
```
int read_id (
    char* buf,              // id string
    int buflen              // length of buffer
        = 0
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
```

Description: The save identifier written with C printf format "%s ". Read an entity identifier. In text mode, this is just a sequence of non–blank characters. In binary mode, it is a sequence of counted strings, of which all but the last have negative counts. These strings are assembled into the buffer, separated by '–'. The result is placed in a caller–supplied buffer – overflow causes an error, unless the length is given zero or negative, in which case no overflow is detected. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_id(buf, buflen) : 0;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_int

Function: SAT Save and Restore

Action: Reads an integer by reading a long and converting.

Prototype:
```
int read_int ();

int read_int(
    const char*& test_string// string to be read
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file. Reads an integer by reading a long and converting. Some compilers will give a warning for this shortening, but it may be ignored. Implementations for machines with ints and longs different lengths may well want a different version. ActiveFile is a FileInterface object and does most of the actual work.

```
return ActiveFile ? (int)(ActiveFile–>read_long()) : 0;
                                        Call the appropriate SatFile or
                                        SabFile method
```

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_interval

Function: SAT Save and Restore

Action: Reads an interval as two doubles.

Prototype:
```
SPAinterval read_interval ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
#include "baseutil/vector/interval.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file. Reads an interval as two doubles (old–style), or as two instances of "I" for infinite, or as "F <value>" for finite bound.

```
if (restore_version_number < INFINT_VERSION)
    read_real                   starting
    read_real                   ending
else
    read_logical                finite: either "I" or "F"
    if (finite)
        read_real               ending
```

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_logical

Function: SAT Save and Restore

Action: Reads a logical.

Prototype:
```
logical read_logical (
    char const* false_str    // string for FALSE
        = ”F”,
    char const* true_str     // string for TRUE
        = ”T”
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernutil/fileio/fileio.hxx”
#include ”baseutil/logical.h”
```

Description: *(false_string, true_string, {or any_valid_string})*: Appropriate string written with C printf format “%s ”. Reads a logical value. Up to LOGICAL_VERSION, this was an integer 0 or 1. Later than that in text files it has been keywords defaulting to ”T” or ”F”. For generality, accept an integer value or any blank–terminated string starting with the first character of either of the given strings. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_logical(false_str, true_str) : FALSE;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_long

Function: SAT Save and Restore

Action: Reads a long written with C printf format “%ld”.

Prototype:
```
long read_long ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernutil/fileio/fileio.hxx”
```

Description: This routine is used as part of restore from a SAT or SAB file. Reads a long integer. In text mode, this ignores initial white space, and leaves the input stream positioned at the character (which should be white space) which terminates the decimal integer representation. In binary, this simply reads the correct number of bytes for the internal representation, and then possibly reorders them. ActiveFile is a FileInterface object and does most of the actual work.

```
return ActiveFile ? ActiveFile–>read_long() : 0;
```

Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_matrix

Function: SAT Save and Restore, Mathematics

Action: Reads a SPAmatrix as three row vectors.

Prototype:
```
SPAmatrix read_matrix ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
#include "baseutil/vector/matrix.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file.

read_vector	vector v1
read_vector	vector v2
read_vector	vector v3

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_pointer

Function: SAT Save and Restore

Action: Reads a pointer.

Prototype:
```
void* read_pointer ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kernutil/fileio/fileio.hxx″
```

Description: Reads a pointer. Pointer reference to a save file record index. Written as “$” followed by index number written as a long. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_pointer() : NULL;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_position

Function: SAT Save and Restore

Action: Reads a position as three doubles.

Prototype:
```
SPAposition read_position ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kernutil/fileio/fileio.hxx″
#include ″baseutil/vector/position.hxx″
```

Description: This routine is used as part of restore from a SAT or SAB file. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_position() : SPAposition(0,0,0);
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_ptr

Function: SAT Save and Restore

Action: Reads a pointer for the save file.

Prototype:
```
ENTITY* read_ptr ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kerndata/savres/savres_small.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file.

return (ENTITY *)read_pointer();	Call the other read pointer function.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kerndata/savres/savres_small.hxx

Effect: Read–only

read_real

Function: SAT Save and Restore

Action: Reads a double.

Prototype:
```
double read_real ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file. Read a double. In text mode, this ignores initial white space, and leaves the input stream positioned at the character (which should be white space) which terminates the decimal representation, which may be fixed–point or exponent notation. In binary, this simply reads the correct number of bytes for the internal representation, and then possibly reorders them. ActiveFile is a FileInterface object and does most of the actual work.

```
return ActiveFile ? ActiveFile–>read_double() : 0;
```

Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_sequence

Function: SAT Save and Restore

Action: Reads an explicit record sequence number.

Prototype:
```
int read_sequence ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kernutil/fileio/fileio.hxx″
```

Description: This routine is used as part of restore from a SAT or SAB file. Reads an explicit record sequence number, returning it, or negative if none. Sequence numbers in text mode consist of a minus sign with no preceding white space, followed by a positive or zero integer. They do not appear in binary files. ActiveFile is a FileInterface object and does most of the actual work.

```
return ActiveFile ? ActiveFile–>read_sequence() : –1;
```

Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_string

Function: SAT Save and Restore

Action: Reads a string into a supplied buffer of a given size, maxlen.

Prototype:

```
char* read_string (
    int& len                // length of buffer
    );

int read_string (
    char* buf               // character string
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file. Reads a string. This consists of an integer length, followed by that number of literal characters. In text mode, the length and characters are separated by exactly one space. In int read_string, we assume that the buffer supplied is of sufficient length for the characters plus the usual terminating null. The function returns the actual number of characters read. The char* read_string is a more convenient form of read_string. The string is written the same as it was for the old version, with a count followed by the actual string. Unlike the old version however, this version allocates a string of the correct length and returns a pointer to it, so you do not have to worry about reading the count, and then backspacing the file to re–read the string if you want to make sure that you have a buffer which is big enough. If the length of the string was zero characters, then this will return NULL rather than "". ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_string(buf) : 0;
Call the appropriate SatFile or SabFile method

return ActiveFile ? ActiveFile–>read_string(len) : NULL;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_subtype_end

Function: SAT Save and Restore

Action: Reads subtype end braces around the subtypes, written as "} ".

Prototype:
```
logical read_subtype_end ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
#include "baseutil/logical.h"
```

Description: This routine is used as part of restore from a SAT or SAB file. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_subtype_end() : FALSE;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_subtype_start

Function: SAT Save and Restore

Action: Reads subtype start braces around the subtypes, written as "{ ".

Prototype:
```
logical read_subtype_start ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
#include "baseutil/logical.h"
```

Description: This routine is used as part of restore from a SAT or SAB file. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_subtype_start() : FALSE;
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_transf

Function: SAT Save and Restore, Mathematics, Transforms

Action: Internal to ACIS and not intended for direct usage. Reads a transformation.

Prototype:
```
SPAtransf read_transf ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
#include "baseutil/vector/transf.hxx"
```

Description: Although this ***internal function is intended strictly for*** ACIS usage, a minimal amount of information about this function is provided for the sole purpose of being able to understand and trace restoration from a SAT file. This function should never be called directly, because it makes assumptions about the availability of a SAT file, the location of the input pointer into the SAT file, and the validity of SAT data it expects to read in. It also may start a lengthy process of nested function or class method calls, which have many of the same assumptions.

Read a transformation as matrix, translation vector, double scaling factor and three integer flags.

	read_matrix	Affine matrix
	read_vector	Translation vector
	read_real	Scaling
	read_logical	Either “no_rotate” or “rotate”
	read_logical	Either “no_reflect” or “reflect”
	read_logical	Either “no_shear” or “shear”

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_unit_vector

Function: SAT Save and Restore

Action: Reads a unit vector as a vector and then normalizes it.

Prototype:
```
SPAunit_vector read_unit_vector ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernutil/fileio/fileio.hxx”
#include ”baseutil/vector/unitvec.hxx”
```

Description: This routine is used as part of restore from a SAT or SAB file. Reads a unit vector as a vector and then normalizes it.

read_vector	Vector to read in.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

read_vector

Function: SAT Save and Restore

Action: Reads a vector as three doubles.

Prototype:
```
SPAvector read_vector ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/fileio/fileio.hxx"
#include "baseutil/vector/vector.hxx"
```

Description: This routine is used as part of restore from a SAT or SAB file. ActiveFile is a FileInterface object and does most of the actual work.

return ActiveFile ? ActiveFile–>read_vector() : SPAvector(0,0,0);
Call the appropriate SatFile or SabFile method

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kernutil/fileio/fileio.hxx

Effect: Read–only

reset_boxes

Function: Construction Geometry

Action: Resets the boxes used.

Prototype:
```
void reset_boxes (
    ENTITY* this_entity     // entity to reset
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kerndata/geometry/getbox.hxx"
```

Description: Resets the bounding box of the topological entity (i.e., set it to NULL), then do the same for its owners. Its argument must be a BODY, LUMP, SHELL, SUBSHELL, FACE, LOOP, EDGE, or WIRE.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kerndata/geometry/getbox.hxx

Effect: Read–only

reset_boxes_downward

Function: Construction Geometry

Action: Resets the box of the given entity and then resets the boxes off all constituent boxes.

Prototype:
```
void reset_boxes_downward (
    ENTITY* ent             // entity to reset
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kerndata/data/entity.hxx″
#include ″kernel/kerndata/geometry/getbox.hxx″
```

Description: Resets the box of the given entity and then resets the boxes off all constituent boxes. In other words, it sets the box pointer to NULL for the portion of the topological tree below this entity.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kerndata/geometry/getbox.hxx

Effect: Read–only

restore_curve

Function: SAT Save and Restore

Action: Internal to ACIS and not intended for direct usage.

Prototype:
```
curve* restore_curve ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kerngeom/curve/curdef.hxx″
```

Description: Although this ***internal function is intended strictly for*** ACIS usage, a minimal amount of information about this function is provided for the sole purpose of being able to understand and trace restoration from a SAT file. This function should never be called directly, because it makes assumptions about the availability of a SAT file, the location of the input pointer into the SAT file, and the validity of SAT data it expects to read in. It also may start a lengthy process of nested function or class method calls, which have many of the same assumptions.

Restores the curve. The restore function does the actual work. It calls the base class, then reads the selector, if the save file is new enough. This reads the curve type and then switches in the run–time table to the correct restore routine.

```
if (restore_version_number < CURVE_VERSION)
    read_int                    integer for the type of curve.
    dispatch_restore_cu         Supply the number for the type of
                                curve
else
    read_id                     Reads in the string associated with
                                the curve identification.
    dispatch_restore_cu         Supply the curve identification for
                                the type of curve
```

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/kerngeom/curve/curdef.hxx

Effect: Read–only