PDS TOOLBOX
OVERVIEW

J. Steve Hughes

June 6-10, 1994

PDS Toolbox

- Label **Creation**
  - tab2lab - Table to Label
  - file2lab - File to Label
- **Label Verification**
  - lvtool - Label Verifier
  - slvtool - Summary, Label verifier
- **Label Access/Write Routines**
  - Label Library - L2
  - Label Library Light - L3
- **Data Display**
  - tbtool - Table Browser

PDS Toolbox

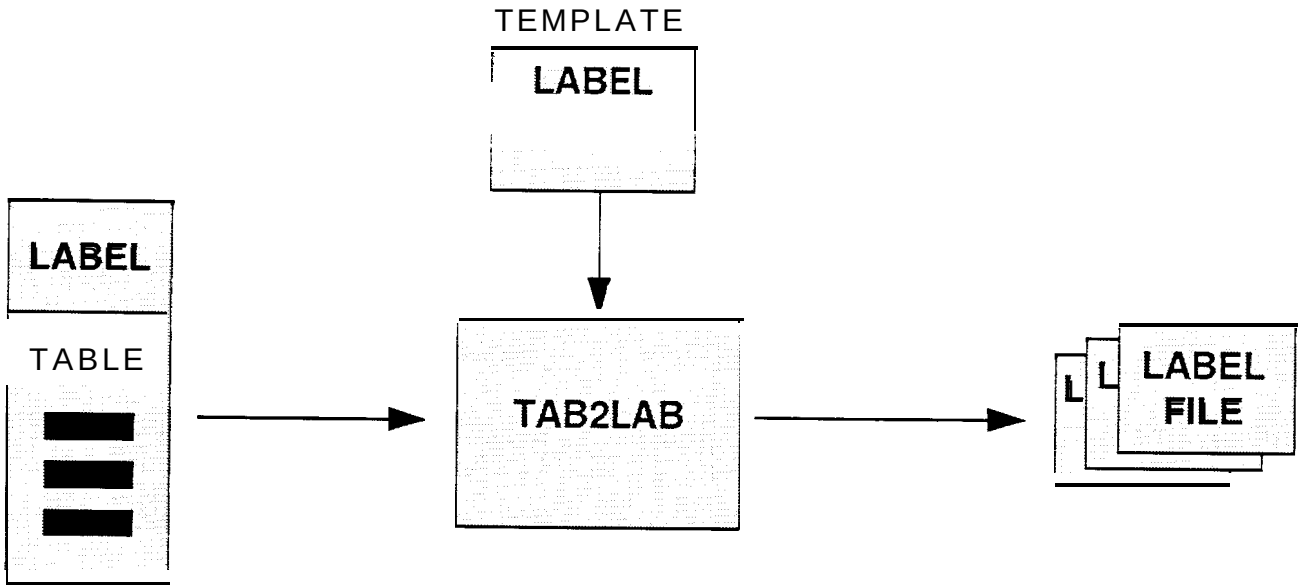
- **Volume Verification**
  - vv- Volume Verifier
- **Volume Index Creation**
  - IGTOOL - Index Generator
- **Other Tools**
  - sled - Simple Label Editor
  - explab - Expand Label
  - make_index - Make Data Dictionary Index
  - ddtool - Customize Data Dictionary Tool
  - labform - Label Formatter

PDS Toolbox

Label Creation - tab2lab

- **PDS Table to Label Generator (tab2lab), Version 2.1**

  A tool for generating PDS labels given a template label and an ASCII table of data values for insertion into the template.



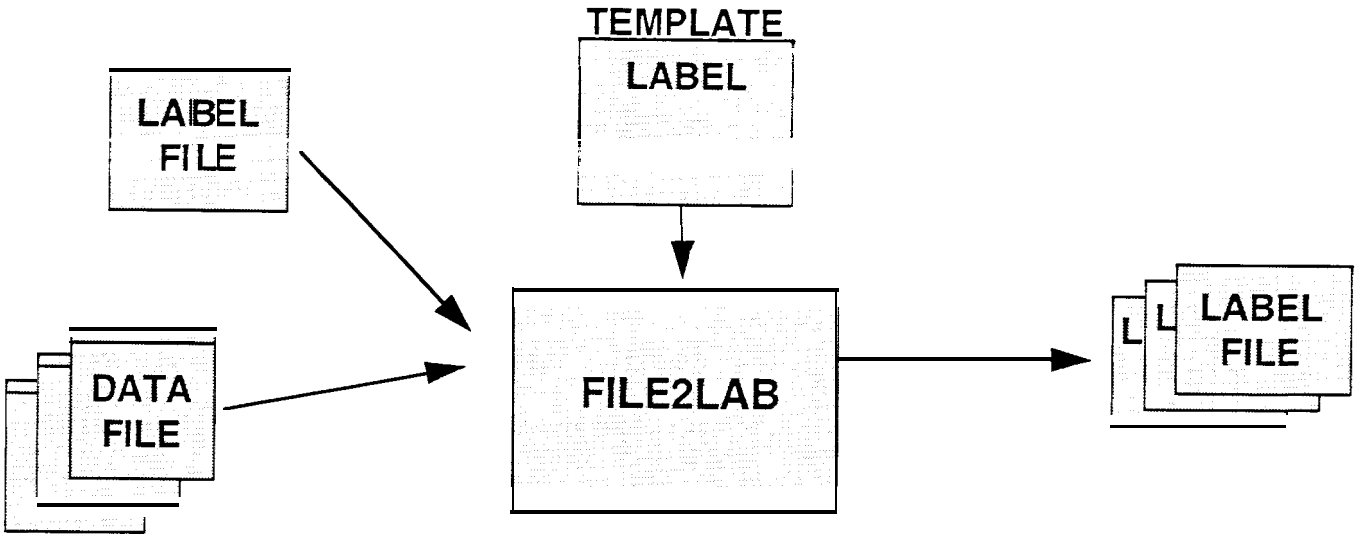
TAB2LAB - READS A PDS TABLE AND CREATES LABEL FILES FOR EACH ROW, USING A TEMPLATE LABEL AS A MODEL.

PDS Toolbox

Label Creation - file2lab

- **PDS File to Label Generator (file2lab), Version BETA**

  A tool for generating PDS labels given a template label and a set of ASCII data files (or file headers) that contain values for insertion into the template.



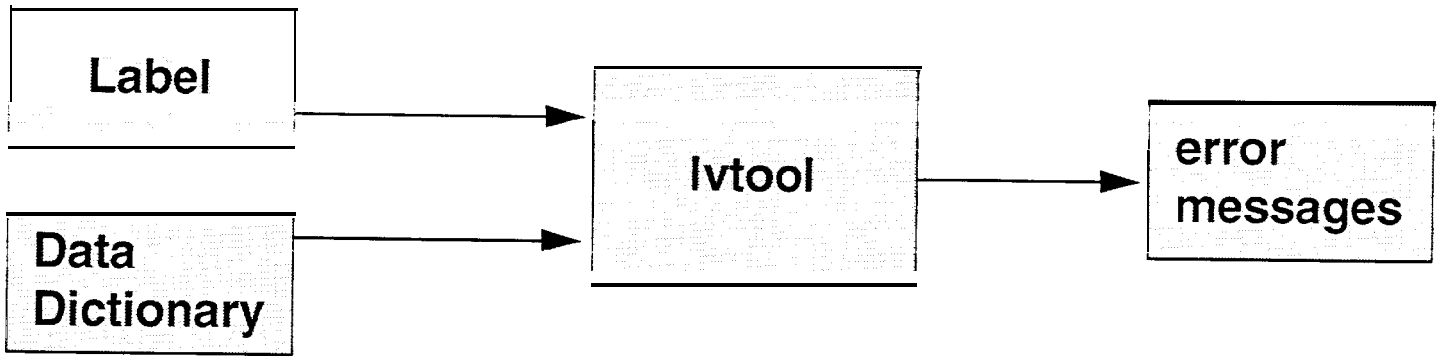
FILE2LAB- CREATES DETACHED LABELS FOR DATA FILES WITH OTHER LABEL FORMATS (FITS, VICAR, ETC). USES TEMPLATE LABEL AS MODEL FOR LABELS.

PDS Toolbox

Label verification - lvtool / slvtool

- **PDS Label Verifier (lvtool), Version 1.2**

  A tool for checking the syntax and semantics of PDS labels by comparing them to the Toolbox Data Dictionary.

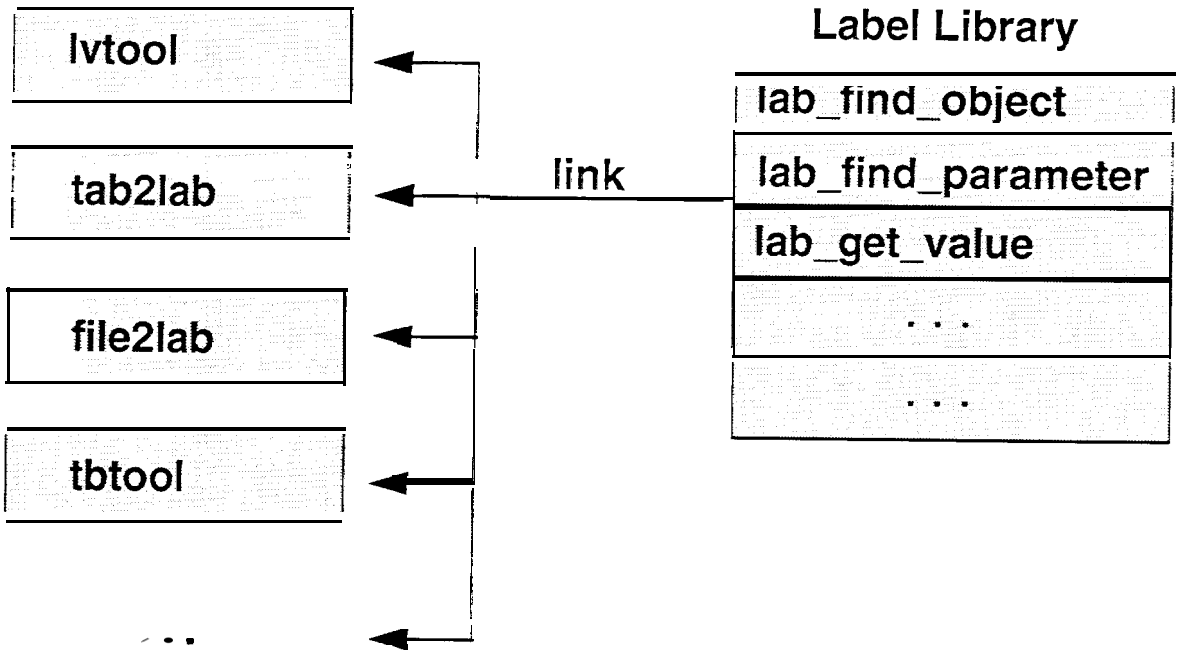


PDS Toolbox

Label Access / Write Routines - L2

- **PDS Label Library (L2) Version 2.0**

  A library of routines for manipulating ODL labels.

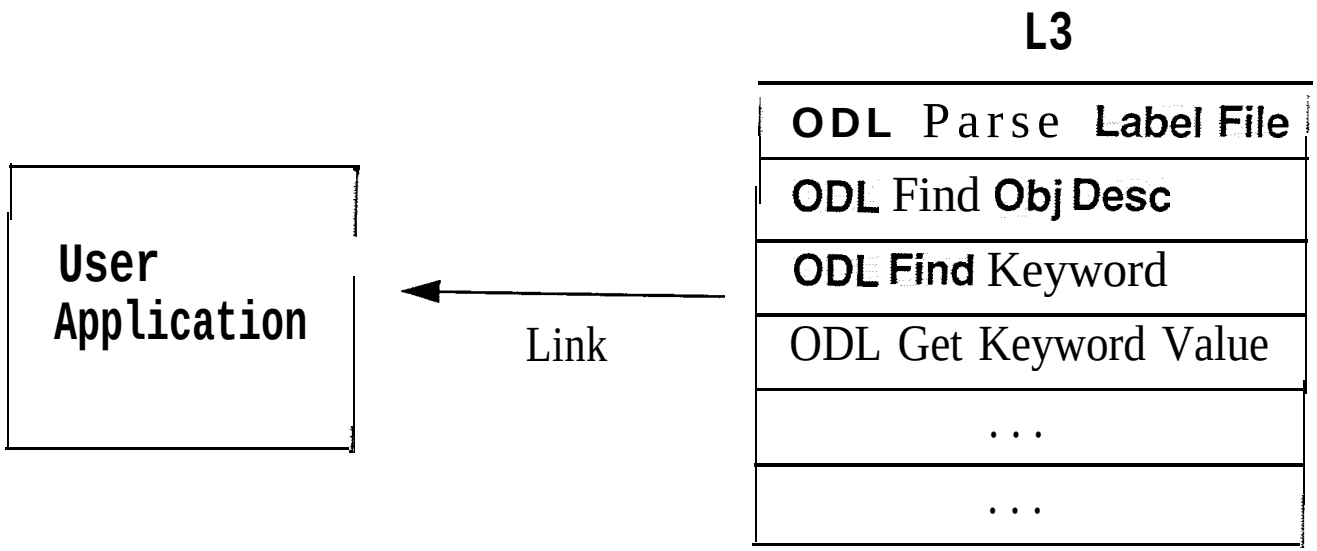


PDS Toolbox

Label Access / Write Routine · L3

- PDS Label Library Light **(L3),** Version 1.0
  Release April 1994

  A set of C functions, to be used for parsing and accessing objects, keywords, and keyword values from PDS labels.

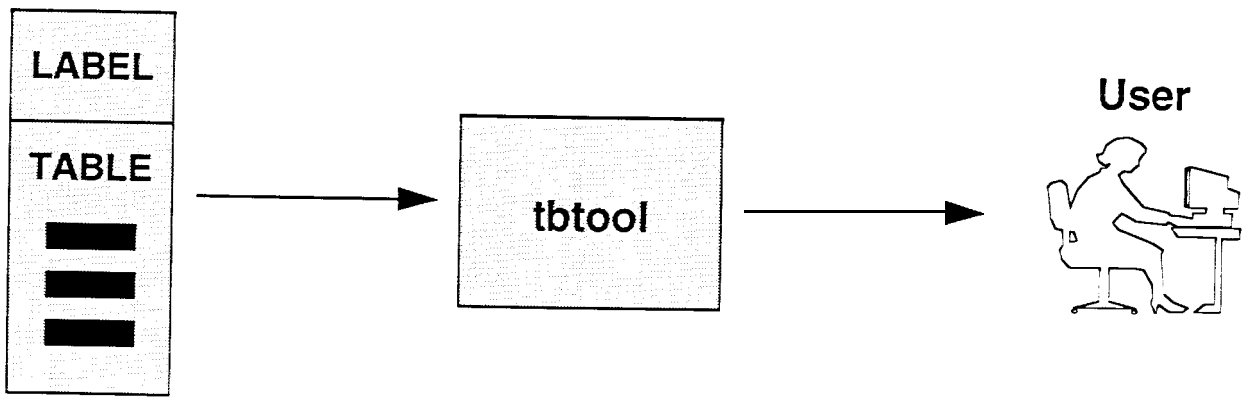


PDS Toolbox

Data Display - tbtool

- **PDS Table Browser (tbtool), Version BETA**

  A tool for interactively examining PDS labelled ASCII and binary data in the form of TABLEs, SERIES, or SPECTRUMs. Supports verification of the correctness of the label, summerization, and visual verification of columns of data.

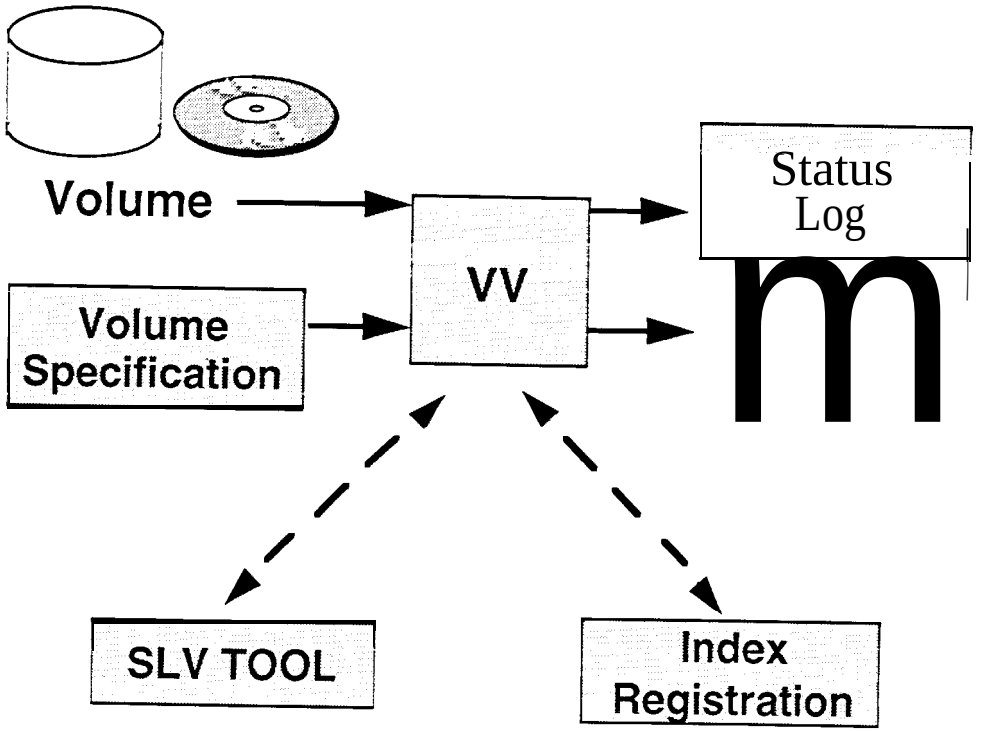


PDS Toolbox

Volume Verifier - vv

- **PDS Volume Verifier (vv), Version** Beta 2.3

  Verifies volume organization, files and standards conformance for volumes on disks, CD-WOS and CD-ROMs.



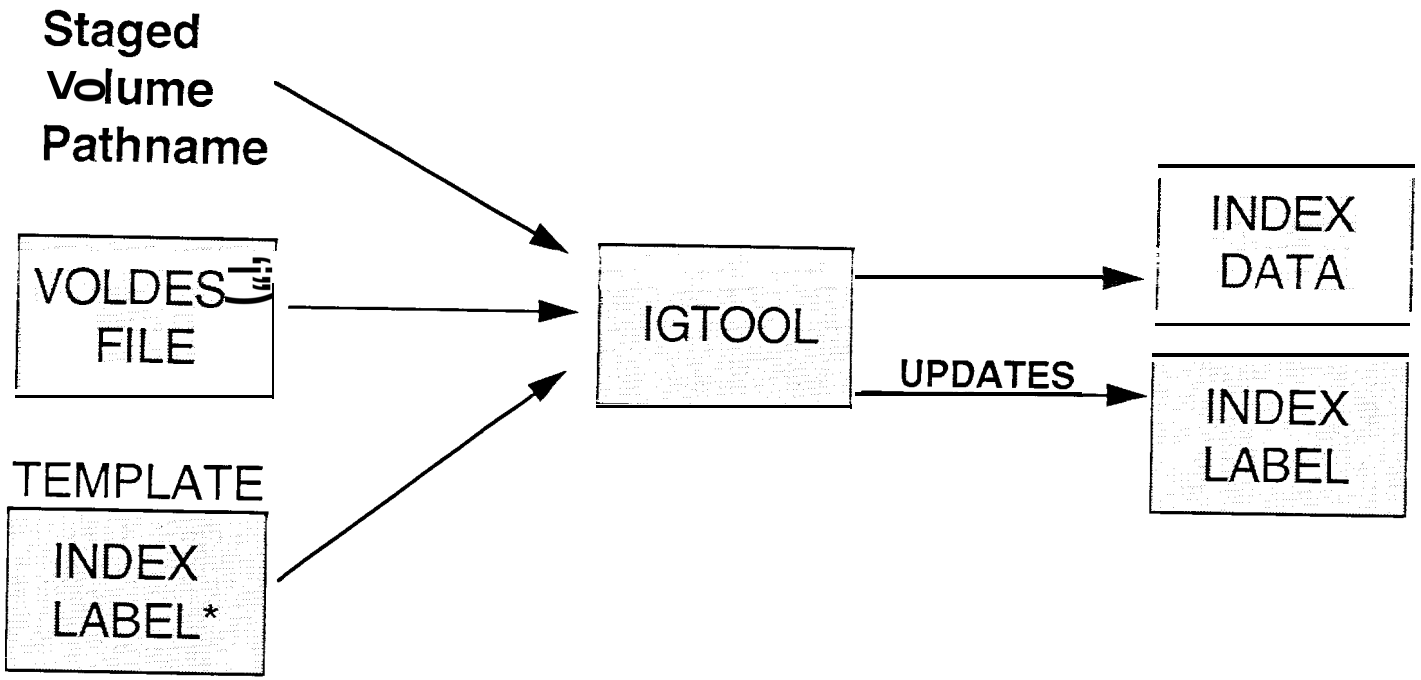
- Missing required or found extra directories/files
- Open / Read
- Maximum nested directory levels
- Zero length files
- Maximum files per directory

--- Optional checks

PDS Toolbox
Volume Index Creation - IGTOOL

- **PDS Index Generator (IGTOOL), Version Alpha**

  A PDS tool to generate an index file for a staged volume.



* INDEX_TYPE
INDEX_SOURCE_FILE_NAME
COLUMN NAMES

PDS Toolbox
Other Tools

- **PDS Simple Label Editor (sled), Version 2.4**

  A command line editor for verifying, modifying, and reformatting PDS labels. Include hooks to the PDS Label Verifier.

- **PDS Label Expander (explab)**

  A tool for expanding a label with STRUCTURE keywords into a new label. (Version 1.0)

- **PDS make Data Dictionary Index (make_index)**

  A tool for generating the index associated with the Toolbox Data Dictionary. (Version 1.2)

- **PDS Customize Data Dictionary (ddtool)**

  A tool for creating a customized data dictionary by selecting items from the PSDD.

- **PDS Label Formatter (labform)**

  A tool for pretty-printing labels. (Version 1.0)

```
---GALILEO IMAGE FILE LABEL - FILE_NAME = 2000 R. LBL---

CCSD3ZFOOO0100000001 NJPL31FOPDS2OOOOOOO1 = SFDU LABEL

      RECORD TYPE = FIXED_LENGTH
      RECORD_BYTES = 1000
      FILE_RECORDS = 814

      'TELEMETRY_TABLE = ("2000 R..IMG",3)
      *IMAGE = ("2000R.IMG",15)
            ...

      DATA_SET_ID = "Go-A/E-s sl-2-REDR-vl. o"
      SPACECRAFT NAME = "GALILEO ORBITER"
      INSTRUMENT-NAME = SOLID_STATE_IMAGING

/"' Time tags and observation descriptors */
      SPACECRAFT_CLOCK_START_COUNT = "01651920.00"
      IMAGE_TIME = 1992-12-09T07:13:01.011Z
      IMAGE_ID = E2W0914
      ORBIT_NUMBER = 12869
            ...

/* Table Object (for telemetry table) */
      OBJECT = TELEMETRY_TABLE
            INTERCHANGE FORMAT = BINARY
            ROWS    =    1
            COLUMNS = 86
            ROW BYTES = 1800
            "STRUCTURE = "RTLMTAB. FMT"
      END_OBJECT

/* Image Object */
      OBJECT = IMAGE
            LINES = 800
            LINE_SAMPLES = 800
            SAMPLE_BITS = 8
            ...
      END_OBJECT
```

```
---GALILEO I-ABLE FORMAT FILE - 'STRUCTURE =
"RTLMTAB.FMT"---

     OBJECT = COLUMN
          NAME = RECORD_ID
          DATA_TYPE = UNSIGNED_INTEGER
          START_BYTE = 1
          BYTES = 1
          DESCRIPTION = "IS always O for the telemetry record"
     END_OBJECT

     OBJECT = COLUMN
          NAME = FILE_NUMBER
          DATA TYPE = UNSIGNED_INTEGER
          START_BYTE = 2
          BYTES = 1
          DESCRIPTION = "Tape file number. Not applicable for
                         CD-ROMs."
     END_OBJECT

     OBJECT = COLUMN
          NAME = MISSION NAME
          DATA_TYPE = CHARACTER
          START_BYTE = 3
          BYTES = 10
          DESCRIPTION = "Mission name, valid is GALI LEO."
     END_OBJECT
          ...
```

----EXAMPLES OF FUNCTION CAILS---

Assume:

```
ODLTREE *label_ptr, *table_ptr, *column_ptr,
        *name_ptr, *image_ptr, *first_ptr;
char name[31], value[11];
int value_type;
```

1) Parse the file, expand the label, and create a new error message file.

```
label_ptr = OdlParseLabelFile
     ("2000R.LBL",">error.lst", EXPAND);
```

2) Find the TELEMETRY_TABLE object description.

```
table_ptr = OdlFindObjDesc (label_ptr, "*TABLE",
                  0, 0, 0, RECURSIVEDOWN);
```

3) Find the second column (FILE_NUMBER) in the table.

```
column_ptr = OdlFindObjDesc (table_ptr, "COLUMN",
                  O, 0, 2, CHILDONLY);
```

4) Find the **NAME** keyword of the column.

```
name_ptr = Odl Find Keyword (column ptr, "NAME",
                  0, 0, OBJECTONLY);
```

5) Get the name (i.e. FILE_NUMBER) of the column.

```
strcpy (name, OdlGet KeywordValue (name_ptr));
```

6) Find the next column (i.e. MISSION_NAME) in the table.

```
column_ptr = OdlFindObjDesc (column ptr, "COLUMN",
                             O, 0, 1, TRAVERSE% IDLINGS);
if (column_ptr != NULL)
```

7) Find the IMAGE object description.

```
image_ptr = OdlFindObjDesc (label ptr, "IMAGE",
                            0, 0, 0, RECURSIVE% OWN);
```

8) Find the first attribute of IMAGE. (i.e. LINES)

```
first_ptr = OdlFind Keyword (image_ptr,
                             O, 0, 1., OBJECTONLY);
if (first_ptr != NULL)
```

9) Get the name, value type, and value of the first keyword of **IMAGE.** (i.e. **LINES, O** (Scalar), **800)**

```
strcpy (name, OdlGetKeyword Name (first_ptr));
value_type = OdlGetKeywordValueTy pe (first_ptr);
strcpy (value, OdlGetKeywordValue (first_ptr));
```

---EXAMPLES OF FUNCTION CALLS---

Assume:

```
ODLTREE *label_ptr, *table_ptr, *image_ptr;
char *image_buffer, *table_buffer;
int image_size, table_size, result;
```

Find the IMAGE object description.

```
image_ptr = OdlFindObjDesc
        (label_ptr, "IMAGE", 0, 0, 0, O);
```

Get the size and pointer to the IMAGE.

```
image_size = OdlGetObjectSize (image_ptr);
```

Get the IMAGE. (after malloc)

```
result = OdlGetObject (image_buffer, image_ptr,
      image_size);
```