PIBASE code guide. ver 2010

Fred P. Davis, HHMI-JFRC
davisf@janelia.hhmi.org

September 19, 2010

Abstract

This document describes the layout of the PIBASE software package. The documentation for all routines is collected here.

1 Overview

The code used to build and access the database is packaged in (1) a perl library, pibase.pm, (2) a caller script build_pibase.pl that calls the perl routines in the order necessary to build the database, and (3) a set of auxiliary programs, written in C and perl, that mainly perform PDB file operations.

The goal of this manual is to describe the code in sufficient detail to allow you to design custom queries through the perl interface. Here, I first describe the (1) code layout in pibase.pm, (2) document some of the routines. The routines themselves have inline pod documentation. For now, this document just collects all the inline documentation into a single file.

```
perl_api/
|-- LGL.pm
|-- pibase
|   |-- ASTRAL.pm
|   |-- CATH.pm
|   |-- PDB
|   |   |-- chains.pm
|   |   |-- residues.pm
|   |   |-- sec_strx.pm
|   |   `-- subsets.pm
|   |-- PDB.pm
```

```
|   |-- PIBASE_core.strx.sql
|   |-- PQS.pm
|   |-- SCOP.pm
|   |-- SGE.pm
|   |-- auxil.pm
|   |-- benchmark.pm
|   |-- build.pm
|   |-- calc
|   |   `-- interfaces.pm
|   |-- create_raw_table_specs.pm
|   |-- data
|   |   |-- access.pm
|   |   |-- calc.pm
|   |   `-- external
|   |       |-- ASTRAL.pm
|   |       `-- PQS.pm
|   |-- data.pm
|   |-- interatomic_contacts.pm
|   |-- kdcontacts.pm
|   |-- modeller.pm
|   |-- pilig.pm
|   |-- raw_table_specs.pm
|   |-- residue_math.pm
|   |-- specs.pm
|   |-- tables_on_disk.pm
|   `-- web.pm
`-- pibase.pm

5 directories, 32 files
```

2 LGL.pm

Perl interface to Alex Adai's Large Graph Layout (LGL) program

DESCRIPTION

The LGL.pm package provides a perl interface to LGL so that edge/node lists can be converted into postscript or png images with user-configurable edge and node colors, shapes, and sizes.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SYNOPSIS

Example 1

```
my $edges = {
   a => { 'b' => 1} ,
   a => { 'd' => 1} ,
   a => { 'e' => 1} ,
   b => { 'c' => 1} ,
   c => { 'e' => 1} ,
} ;
my $coords = LGL::edges2coords({edges => $edges}) ;
coords2ps({
   edges => $edges,
   coords => $coords,
}) ;
```

Example 2

```
my $edges ;
$edges->{a}->{b} = 1 ;
$edges->{a}->{c} = 1 ;
$edges->{b}->{d} = 1 ;

my $ncols;
$ncols->{"a"} = "black" ;
$ncols->{"b"} = "yellow";
$ncols->{"c"} = "cyan";
$ncols->{"d"} = "red";

my $ecols ;
$ecols->{a}->{b} = "black" ;
$ecols->{a}->{c} = "purple" ;
$ecols->{b}->{d} = "brown" ;
```

```
my $coords = LGL::edges2coords({edges => $edges}) ;
LGL::coords2ps({coords => $coords,
                edges => $edges,
                mag => 100,
                ethick => 1,
                nrad => 5,
                nshape => 'fbox',
                ncol => "0.5 0.3 0.2",
                ncols => $ncols,
                ecols => $ecols,
                ecol => "0.2 0.7 0.4"}) ;
```

SUBROUTINES

edges2coords()

```
Title:       edges2coords()
Function:    Performs graph layout using LGL, returning coordinate info
Args:        $_->{edges}->{node1}->{node2} ;
Returns:     $_->{node} = [$x, $y] ;
```

_make_lglconfig()

```
Title:       _make_lglconfig
Function:    Makes a config file for an LGL run.
Returns:     nothing
Args:        $_->{cnfg_fh} (file handle to display the configuration file to)
```

coords2pngmap()

```
Title:       coords2pngmap
Function:    Creates a client-side png image map given coordinate and
               edge information.
Returns:     nothing
Args:
   - $_->{ncol}      node color
   - $_->{ecol}      edge color
   - $_->{estyle}    edge style (solid, dashed)
   - $_->{nshape}    node shape (triangle, square, circle)
   - $_->{nrad}      node radius
   - $_->{ethick}    edge thickness
   - $_->{mag}       magnification parameter
   - $_->{coords}    node coordinate information
   - $_->{edges}->{n1}->{n2} edge list (nhash)
   - $_->{map_fh}    output imagemap file name
   - $_->{nodenames} node names
   - $_->{nodeurls}  node URLs
```

coords2png()

```
Title:       coords2png
Function:    Creates a png image of a graph, given layout info
Returns:     GD::image object
Args:
   $_->{coords}      node coordinate information
   $_->{edges}->{n1}->{n2}   edge list (nhash)
   $_->{ncol}        default node color
   $_->{ncols}->{n}  node-specific color
   $_->{ecol}        default edge color
   $_->{ecols}->{n1}->{n2}   edge-specific color
   $_->{estyle}      default edge style - solid or dashed
   $_->{estyles}->{n1}->{n2} edge-specific style
   $_->{nshape}      node shape ([f]{triangle|box|circle})
   $_->{nshapes}->{n}        node-specific shape
   $_->{nrad}        node radius
   $_->{ethick}      edge thickness
   $_->{mag} magnification parameter
```

coords2ps()

```
Title:       coords2ps
Function:    Creates a postscript image of a graph, given layout info
Returns:     nothing
Args:
   $_->{coords}      node coordinate information
   $_->{edges}->{n1}->{n2}   edge list (nhash)
   $_->{ncol}        default node color
   $_->{ncols}->{n}  node-specific color
   $_->{ecol}        default edge color
   $_->{ecols}->{n1}->{n2}   edge-specific color
   $_->{estyle}      default edge style - solid or dashed
   $_->{estyles}->{n1}->{n2} edge-specific style
   $_->{nshape}      node shape ([f]{triangle|box|circle})
   $_->{nshapes}->{n}        node-specific shape
   $_->{nrad}        node radius
   $_->{ethick}      edge thickness
   $_->{mag} magnification parameter
```

get_color2rgb()

```
Title:       get_color2rgb()
Function:    Returns a hash pointing from color name to rgb (0-1) values
Args:        Nothing
Returns:     $_->{color} = "$r $g $b" ;
```

3 pibase.pm

Perl interface to the pibase database

DESCRIPTION

The pibase.pm perl library contains subroutines used to build and access the PIBASE database.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

dbname()

```
Title:      dbname()
Function:   gets the name of the pibase database
Args:       pibase $specs data structure
Returns:    returns the name of the pibase mysql database
```

get_specs()

```
Title:      get_specs()
Function:   gives pibase specifications
Args:       pibase $specs data structure
Returns:    returns completed pibase data specifications
```

connect_pibase($dbspecs)

```
Title:      connect_pibase()
Function:   Connects to the pibase database.
Args:       $_->{db}        database name
            $_->{user}      user name
            $_->{pass}      password
Returns:    DBI database handle to pibaes
```

connect_tod($dbspecs)

```
Title:      connect_tod()
Function:   Connects to pibase tables on disk (POD).
Args:       $_[0] = tablename
            $_[1] = dbspecs
            $_[2] = tod_dir = location of tables on disk

Returns:    DBI database handle to table on disk
```

rawselect_tod()

```
Title:       rawselect_tod()
Function:    performs basic SELECT statement queries on a table on disk
Args:        $_[0] = SQL SELECT-like statement
             $_[1] = fullfile
```

connect_metatod()

```
Title:       connect_metatod()
Function:    performs basic SELECT statement queries on a meta-table on disk
Args:        $_[0] = filename
             $_[1] = tablename
             $_[2] = pibase db specs
Returns:     dbh DBI:AnyData database handle
```

rawselect_metatod()

```
Title:       rawselect_metatod()
Function:    selects specified fields from a table-on-disk.
   Note: WHERE clause does not work, this command just recognizes
         the field names and returns the appropriate columns.
Args:        $_[0] = filename
             $_[1] = SELECT sql command
Returns:     array of query results
```

sid_2_domdir()

```
Title:       sid_2_domdir()
Function:    returns the directory name where the PDB file of the
   specified domain resides.
Args:        $_ = subset_id
Returns:     directory name
```

complete_pibase_specs(specs)

```
Title:       complete_pibase_specs
Function:    Fills in blanks in specs with default values.
Input:       $_ = specs hashref
             $_->{db} = database_name
             $_->{user} = user name
             $_->{pass} = password
Return:      specs - hashref
```

load_bdp_ids($dbh, @results_type)

```
Name: load_bdp_ids()
```

```
Function:    Returns bdp_id and depending on results_type
             specified, its relation to bdp_path and pdb_id
             in a variety of forms.
Return:      results
Args:        $_[0] = DBI dbh handle
             $_[1] = results type
```

- path_2_bdp_id (hash) [default]
- bdp_id_2_path (hash)
- bdp_id_2_pdb_id (hash)
- bdp_id_2_raw_pdb (hash)
- pdb_id_2_bdp_id (hash)
- bdp_id (array)

todload_bdp_ids(@results_type)

```
Name: todload_bdp_ids()
Function:    Returns bdp_id and depending on results_type
             specified, its relation to bdp_path and pdb_id
             in a variety of forms.
   Analogous to load_bdp_ids() with tables-on-disk instead of DBI
Return:      query results
Args:        $_[0] = DBI dbh handle
             $_[1] = results type
```

- path_2_bdp_id (hash) [default]
- bdp_id_2_path (hash)
- bdp_id_2_pdb_id (hash)
- bdp_id_2_raw_pdb (hash)
- pdb_id_2_bdp_id (hash)
- bdp_id (array)

mysql_fetchcols(dbh, query)

```
Function:    Processes an n column query and returns a list of array references, where ea
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL select query

Returns:     @_ - list of arrayref
             $a[i]->[j] ith column, jth row
```

mysql_hashindload(dbh, query)

```
Name:        mysql_hashindload() ;
Function:    Processes a 1 column query and returns a hash pointing from
               column1 values to row number.
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL SELECT query
             $_[2] = substitutor for undefined value
Returns:     $_ = hashref
             $a->{col1} = row number
```

array2hash(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and returns a hashref with value,
             index pairs.
Args:        $_[0] = array references
             $_[1] = undefined substitution - if the cell contains an
               undefined value, use this value as the hash key
Returns:     $_ = hashref
             $a->{value} = row number
```

replace_undefs(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) undefined
               values to a specified substitution value.
Args:        $_[0] = array references
             $_[1] = undefined substitution - if the cell contains an
               undefined value, replace with this value
Returns:     $_ = array reference
```

replace_undefs_blanks(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) undefined and
               blank values to a specified substitution value.
Args:        $_[0] = array references
             $_[1] = undefined/blank substitution - if the cell
               contains an undefined or blank value, replace with this value
Returns:     $_ = array reference
```

replace_char(arrayref, target, replacement)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) target values to a specifie
Args:        $_[0] = array reference
```

```
              $_[1] = target value
              $_[2] = substitution value
Returns:      $_ = array reference
```

mysql_hashload(dbh, query)

```
Name:         mysql_hashload()
Function:     Processes a 2 column query and returns a hash pointing from
                column1 values to column2 values (use for 1:1 relationships)
Args:         $_[0] = dbh - DBI database handle
              $_[1] = SQL SELECT command
Returns:      $a - hashref
              $a->{col1} = col2
```

mysql_hash2load(dbh, query)

```
Name:         mysql_hash2load()
Function:     Processes a 3 column query and returns a hash pointing from
                column1 values to column2 to column3 values
                (use for 1:1:1 relationships)
Args:         $_[0] = dbh - DBI database handle
              $_[1] = SQL SELECT command
Returns:      $a - hashref
              $a->{col1}->{col2} = col3 ;
```

mysql_hasharrload(dbh, query, undef_subs)

```
Name:         mysql_hashload()
Function:     Processes a 2 column query and returns a hash pointing from
                column1 values to an array of column2 values
                (use for 1:n relationships)
Args:         $_[0] dbh - DBI database handle
              $_[1] query - SQL format
              $_[2] undefined value substitutors - list
Returns:      $a - hashref to arrayrefs
$a->{col1} = [ col2_1, col2_2,... col2_n ]
```

mysql_singleval(dbh, query)

```
Function:     Processes a 1 column, 1 row query and returns a scalar
                containing the value.
Args:         $_[0] = dbh - DBI database handle
              $_[1] = query - SQL format
Return:       $a - scalar
              $a = "result"
```

timestamp()

```
Function:    Returns a timestamp
Args:        none
Return:      $_[0] = timestamp: <4-digit YEAR><2-digit MONTH><2-digit DAY>_
               <2-digit HOUR><2-digit MINUTE>
```

get_current_date_mysql()

```
Function:    Returns current date in YYYY-MM-DD mysql format
Args:        none
Return:      $_[0] = date: <4-digit YEAR>-<2-digit MONTH>-<2-digit DAY>
```

timestampsec()

```
Function:    Returns a second-resolution timestamp
Args:        none
Return:      timestamp: <4-digit YEAR><2-digit MONTH><2-digit DAY>_
               <2-digit HOUR><2-digit MINUTE><2-digit SECOND>
```

mysqlimport(file,dbspecs)

```
Function:    load a file into a mysql database using system(mysqlimport)
Return:      $_[0] = records imported
             $_[1] = records deleted
             $_[2] = records skipped
             $_[3] = number of warnings

Args:        $_[0] = filename
             $_[1] = database specs - hashref
```

- db => database name
- user => user name
- pass => password

mysql_runcom(dbh, query, vaues)

```
Function:    Runs an non-SELECT SQL statement
Return:      none
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL query command or DBI statement handle
             $_[2] = values- arrayref that hold values for '?' holders
                     in DBI statement handle
```

mysql_createtable(dbh, tablename, spec)

```
Function:    Creates a table (NOTE: if table already exists, drops it first).
Return:      none
Args:        $_[0] = dbh - DBI database handle
             $_[1] = table name
             $_[2] = spec - SQL DDL string

Example uasge:

   pibase::mysql_createtable($dbh,
      $tables->{interface_contacts}->{name},
      $tables->{interface_contacts}->{spec}) ;

   pibase::mysql_runcom($dbh,
      "REPLACE INTO $tables->{interface_contacts}->{meta} ".
      "( bdp_id, table_name) values($bdp_id, ".
      "\"$tables->{interface_contacts}->{name}\")") ;
```

mysql_commandline_query(dbh, query, vaues)

```
Function:    Runs an SQL SELECT statement on the command line,
             optionally performs a sort (on command line), and
             displays the output to a specified file
Return:      none
Args:        $->{db_name} = database name
             $->{sql} = SQL query
             $->{out_fn} = file to display results to
             $->{post_sort} = optional specify field to sort
             $->{sort_order} = optional sort order (defaults ASC)
```

locate_binaries()

```
Function:    Returns location of binaries used in PIBASE associated activities
Return:      $_->{program} = program location.
   perl, zcat, rigor, subset_extractor, altloc_check
Args:        none
```

safe_move()

```
Function:    Safely move a file to a directory (using File::Copy::move),
               retries 14 times, and prints an error if it didnt work
Return:      nothing
Args:        $_[0] = source filename
             $_[1] = target directory
```

safe_copy()

```
Function:    Safely copy a file to a directory (using File::Copy::copy),
               retries 14 times, and prints an error if it didnt work
Return:      nothing
Args:        $_[0] = source filename
             $_[1] = target directory
```

4 pibase.pm

Perl interface to the pibase database

DESCRIPTION

The pibase.pm perl library contains subroutines used to build and access the PIBASE database.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

dbname()

```
Title:      dbname()
Function:   gets the name of the pibase database
Args:       pibase $specs data structure
Returns:    returns the name of the pibase mysql database
```

get_specs()

```
Title:      get_specs()
Function:   gives pibase specifications
Args:       pibase $specs data structure
Returns:    returns completed pibase data specifications
```

connect_pibase($dbspecs)

```
Title:      connect_pibase()
Function:   Connects to the pibase database.
Args:       $_->{db}        database name
            $_->{user}      user name
            $_->{pass}      password
Returns:    DBI database handle to pibaes
```

connect_tod($dbspecs)

```
Title:      connect_tod()
Function:   Connects to pibase tables on disk (POD).
Args:       $_[0] = tablename
            $_[1] = dbspecs
            $_[2] = tod_dir = location of tables on disk

Returns:    DBI database handle to table on disk
```

rawselect_tod()

```
Title:       rawselect_tod()
Function:    performs basic SELECT statement queries on a table on disk
Args:        $_[0] = SQL SELECT-like statement
             $_[1] = fullfile
```

connect_metatod()

```
Title:       connect_metatod()
Function:    performs basic SELECT statement queries on a meta-table on disk
Args:        $_[0] = filename
             $_[1] = tablename
             $_[2] = pibase db specs
Returns:     dbh DBI:AnyData database handle
```

rawselect_metatod()

```
Title:       rawselect_metatod()
Function:    selects specified fields from a table-on-disk.
   Note: WHERE clause does not work, this command just recognizes
         the field names and returns the appropriate columns.
Args:        $_[0] = filename
             $_[1] = SELECT sql command
Returns:     array of query results
```

sid_2_domdir()

```
Title:       sid_2_domdir()
Function:    returns the directory name where the PDB file of the
   specified domain resides.
Args:        $_ = subset_id
Returns:     directory name
```

complete_pibase_specs(specs)

```
Title:       complete_pibase_specs
Function:    Fills in blanks in specs with default values.
Input:       $_ = specs hashref
             $_->{db} = database_name
             $_->{user} = user name
             $_->{pass} = password
Return:      specs - hashref
```

load_bdp_ids($dbh, @results_type)

```
Name: load_bdp_ids()
```

```
Function:    Returns bdp_id and depending on results_type
             specified, its relation to bdp_path and pdb_id
             in a variety of forms.
Return:      results
Args:        $_[0] = DBI dbh handle
             $_[1] = results type
```

- path_2_bdp_id (hash) [default]
- bdp_id_2_path (hash)
- bdp_id_2_pdb_id (hash)
- bdp_id_2_raw_pdb (hash)
- pdb_id_2_bdp_id (hash)
- bdp_id (array)

todload_bdp_ids(@results_type)

```
Name: todload_bdp_ids()
Function:    Returns bdp_id and depending on results_type
             specified, its relation to bdp_path and pdb_id
             in a variety of forms.
   Analogous to load_bdp_ids() with tables-on-disk instead of DBI
Return:      query results
Args:        $_[0] = DBI dbh handle
             $_[1] = results type
```

- path_2_bdp_id (hash) [default]
- bdp_id_2_path (hash)
- bdp_id_2_pdb_id (hash)
- bdp_id_2_raw_pdb (hash)
- pdb_id_2_bdp_id (hash)
- bdp_id (array)

mysql_fetchcols(dbh, query)

```
Function:    Processes an n column query and returns a list of array references, where ea
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL select query

Returns:     @_ - list of arrayref
             $a[i]->[j] ith column, jth row
```

mysql_hashindload(dbh, query)

```
Name:        mysql_hashindload() ;
Function:    Processes a 1 column query and returns a hash pointing from
               column1 values to row number.
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL SELECT query
             $_[2] = substitutor for undefined value
Returns:     $_ = hashref
             $a->{col1} = row number
```

array2hash(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and returns a hashref with value,
             index pairs.
Args:        $_[0] = array references
             $_[1] = undefined substitution - if the cell contains an
               undefined value, use this value as the hash key
Returns:     $_ = hashref
             $a->{value} = row number
```

replace_undefs(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) undefined
               values to a specified substitution value.
Args:        $_[0] = array references
             $_[1] = undefined substitution - if the cell contains an
               undefined value, replace with this value
Returns:     $_ = array reference
```

replace_undefs_blanks(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) undefined and
               blank values to a specified substitution value.
Args:        $_[0] = array references
             $_[1] = undefined/blank substitution - if the cell
               contains an undefined or blank value, replace with this value
Returns:     $_ = array reference
```

replace_char(arrayref, target, replacement)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) target values to a specifie
Args:        $_[0] = array reference
```

```
              $_[1] = target value
              $_[2] = substitution value
Returns:      $_ = array reference
```

mysql_hashload(dbh, query)

```
Name:         mysql_hashload()
Function:     Processes a 2 column query and returns a hash pointing from
                column1 values to column2 values (use for 1:1 relationships)
Args:         $_[0] = dbh - DBI database handle
              $_[1] = SQL SELECT command
Returns:      $a - hashref
              $a->{col1} = col2
```

mysql_hash2load(dbh, query)

```
Name:         mysql_hash2load()
Function:     Processes a 3 column query and returns a hash pointing from
                column1 values to column2 to column3 values
                (use for 1:1:1 relationships)
Args:         $_[0] = dbh - DBI database handle
              $_[1] = SQL SELECT command
Returns:      $a - hashref
              $a->{col1}->{col2} = col3 ;
```

mysql_hasharrload(dbh, query, undef_subs)

```
Name:         mysql_hashload()
Function:     Processes a 2 column query and returns a hash pointing from
                column1 values to an array of column2 values
                (use for 1:n relationships)
Args:         $_[0] dbh - DBI database handle
              $_[1] query - SQL format
              $_[2] undefined value substitutors - list
Returns:      $a - hashref to arrayrefs
$a->{col1} = [ col2_1, col2_2,... col2_n ]
```

mysql_singleval(dbh, query)

```
Function:     Processes a 1 column, 1 row query and returns a scalar
                containing the value.
Args:         $_[0] = dbh - DBI database handle
              $_[1] = query - SQL format
Return:       $a - scalar
              $a = "result"
```

timestamp()

```
Function:    Returns a timestamp
Args:        none
Return:      $_[0] = timestamp: <4-digit YEAR><2-digit MONTH><2-digit DAY>_
               <2-digit HOUR><2-digit MINUTE>
```

get_current_date_mysql()

```
Function:    Returns current date in YYYY-MM-DD mysql format
Args:        none
Return:      $_[0] = date: <4-digit YEAR>-<2-digit MONTH>-<2-digit DAY>
```

timestampsec()

```
Function:    Returns a second-resolution timestamp
Args:        none
Return:      timestamp: <4-digit YEAR><2-digit MONTH><2-digit DAY>_
               <2-digit HOUR><2-digit MINUTE><2-digit SECOND>
```

mysqlimport(file,dbspecs)

```
Function:    load a file into a mysql database using system(mysqlimport)
Return:      $_[0] = records imported
             $_[1] = records deleted
             $_[2] = records skipped
             $_[3] = number of warnings

Args:        $_[0] = filename
             $_[1] = database specs - hashref
```

- db => database name
- user => user name
- pass => password

mysql_runcom(dbh, query, vaues)

```
Function:    Runs an non-SELECT SQL statement
Return:      none
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL query command or DBI statement handle
             $_[2] = values- arrayref that hold values for '?' holders
                     in DBI statement handle
```

mysql_createtable(dbh, tablename, spec)

```
Function:    Creates a table (NOTE: if table already exists, drops it first).
Return:      none
Args:        $_[0] = dbh - DBI database handle
             $_[1] = table name
             $_[2] = spec - SQL DDL string

Example uasge:

   pibase::mysql_createtable($dbh,
      $tables->{interface_contacts}->{name},
      $tables->{interface_contacts}->{spec}) ;

   pibase::mysql_runcom($dbh,
      "REPLACE INTO $tables->{interface_contacts}->{meta} ".
      "( bdp_id, table_name) values($bdp_id, ".
      "\"$tables->{interface_contacts}->{name}\")") ;
```

mysql_commandline_query(dbh, query, vaues)

```
Function:    Runs an SQL SELECT statement on the command line,
             optionally performs a sort (on command line), and
             displays the output to a specified file
Return:      none
Args:        $->{db_name} = database name
             $->{sql} = SQL query
             $->{out_fn} = file to display results to
             $->{post_sort} = optional specify field to sort
             $->{sort_order} = optional sort order (defaults ASC)
```

locate_binaries()

```
Function:    Returns location of binaries used in PIBASE associated activities
Return:      $_->{program} = program location.
   perl, zcat, rigor, subset_extractor, altloc_check
Args:        none
```

safe_move()

```
Function:    Safely move a file to a directory (using File::Copy::move),
               retries 14 times, and prints an error if it didnt work
Return:      nothing
Args:        $_[0] = source filename
             $_[1] = target directory
```

safe_copy()

```
Function:    Safely copy a file to a directory (using File::Copy::copy),
               retries 14 times, and prints an error if it didnt work
Return:      nothing
Args:        $_[0] = source filename
             $_[1] = target directory
```

5 pibase.pm

Perl interface to the pibase database

DESCRIPTION

The pibase.pm perl library contains subroutines used to build and access the PIBASE database.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

dbname()

```
Title:      dbname()
Function:   gets the name of the pibase database
Args:       pibase $specs data structure
Returns:    returns the name of the pibase mysql database
```

get_specs()

```
Title:      get_specs()
Function:   gives pibase specifications
Args:       pibase $specs data structure
Returns:    returns completed pibase data specifications
```

connect_pibase($dbspecs)

```
Title:      connect_pibase()
Function:   Connects to the pibase database.
Args:       $_->{db}        database name
            $_->{user}      user name
            $_->{pass}      password
Returns:    DBI database handle to pibaes
```

connect_tod($dbspecs)

```
Title:      connect_tod()
Function:   Connects to pibase tables on disk (POD).
Args:       $_[0] = tablename
            $_[1] = dbspecs
            $_[2] = tod_dir = location of tables on disk

Returns:    DBI database handle to table on disk
```

rawselect_tod()

```
Title:       rawselect_tod()
Function:    performs basic SELECT statement queries on a table on disk
Args:        $_[0] = SQL SELECT-like statement
             $_[1] = fullfile
```

connect_metatod()

```
Title:       connect_metatod()
Function:    performs basic SELECT statement queries on a meta-table on disk
Args:        $_[0] = filename
             $_[1] = tablename
             $_[2] = pibase db specs
Returns:     dbh DBI:AnyData database handle
```

rawselect_metatod()

```
Title:       rawselect_metatod()
Function:    selects specified fields from a table-on-disk.
   Note: WHERE clause does not work, this command just recognizes
         the field names and returns the appropriate columns.
Args:        $_[0] = filename
             $_[1] = SELECT sql command
Returns:     array of query results
```

sid_2_domdir()

```
Title:       sid_2_domdir()
Function:    returns the directory name where the PDB file of the
   specified domain resides.
Args:        $_ = subset_id
Returns:     directory name
```

complete_pibase_specs(specs)

```
Title:       complete_pibase_specs
Function:    Fills in blanks in specs with default values.
Input:       $_ = specs hashref
             $_->{db} = database_name
             $_->{user} = user name
             $_->{pass} = password
Return:      specs - hashref
```

load_bdp_ids($dbh, @results_type)

```
Name: load_bdp_ids()
```

```
Function:    Returns bdp_id and depending on results_type
             specified, its relation to bdp_path and pdb_id
             in a variety of forms.
Return:      results
Args:        $_[0] = DBI dbh handle
             $_[1] = results type
```

- path_2_bdp_id (hash) [default]
- bdp_id_2_path (hash)
- bdp_id_2_pdb_id (hash)
- bdp_id_2_raw_pdb (hash)
- pdb_id_2_bdp_id (hash)
- bdp_id (array)

todload_bdp_ids(@results_type)

```
Name: todload_bdp_ids()
Function:    Returns bdp_id and depending on results_type
             specified, its relation to bdp_path and pdb_id
             in a variety of forms.
   Analogous to load_bdp_ids() with tables-on-disk instead of DBI
Return:      query results
Args:        $_[0] = DBI dbh handle
             $_[1] = results type
```

- path_2_bdp_id (hash) [default]
- bdp_id_2_path (hash)
- bdp_id_2_pdb_id (hash)
- bdp_id_2_raw_pdb (hash)
- pdb_id_2_bdp_id (hash)
- bdp_id (array)

mysql_fetchcols(dbh, query)

```
Function:    Processes an n column query and returns a list of array references, where ea
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL select query

Returns:     @_ - list of arrayref
             $a[i]->[j] ith column, jth row
```

mysql_hashindload(dbh, query)

```
Name:        mysql_hashindload() ;
Function:    Processes a 1 column query and returns a hash pointing from
               column1 values to row number.
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL SELECT query
             $_[2] = substitutor for undefined value
Returns:     $_ = hashref
             $a->{col1} = row number
```

array2hash(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and returns a hashref with value,
             index pairs.
Args:        $_[0] = array references
             $_[1] = undefined substitution - if the cell contains an
               undefined value, use this value as the hash key
Returns:     $_ = hashref
             $a->{value} = row number
```

replace_undefs(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) undefined
               values to a specified substitution value.
Args:        $_[0] = array references
             $_[1] = undefined substitution - if the cell contains an
               undefined value, replace with this value
Returns:     $_ = array reference
```

replace_undefs_blanks(arrayref, undef_sub)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) undefined and
               blank values to a specified substitution value.
Args:        $_[0] = array references
             $_[1] = undefined/blank substitution - if the cell
               contains an undefined or blank value, replace with this value
Returns:     $_ = array reference
```

replace_char(arrayref, target, replacement)

```
Name:        array2hash() ;
Function:    Takes an array reference and replaces (inplace) target values to a specifie
Args:        $_[0] = array reference
```

```
              $_[1] = target value
              $_[2] = substitution value
   Returns:   $_ = array reference
```

mysql_hashload(dbh, query)

```
   Name:      mysql_hashload()
   Function:  Processes a 2 column query and returns a hash pointing from
                column1 values to column2 values (use for 1:1 relationships)
   Args:      $_[0] = dbh - DBI database handle
              $_[1] = SQL SELECT command
   Returns:   $a - hashref
              $a->{col1} = col2
```

mysql_hash2load(dbh, query)

```
   Name:      mysql_hash2load()
   Function:  Processes a 3 column query and returns a hash pointing from
                column1 values to column2 to column3 values
                (use for 1:1:1 relationships)
   Args:      $_[0] = dbh - DBI database handle
              $_[1] = SQL SELECT command
   Returns:   $a - hashref
              $a->{col1}->{col2} = col3 ;
```

mysql_hasharrload(dbh, query, undef_subs)

```
   Name:      mysql_hashload()
   Function:  Processes a 2 column query and returns a hash pointing from
                column1 values to an array of column2 values
                (use for 1:n relationships)
   Args:      $_[0] dbh - DBI database handle
              $_[1] query - SQL format
              $_[2] undefined value substitutors - list
   Returns:   $a - hashref to arrayrefs
$a->{col1} = [ col2_1, col2_2,... col2_n ]
```

mysql_singleval(dbh, query)

```
   Function:  Processes a 1 column, 1 row query and returns a scalar
                containing the value.
   Args:      $_[0] = dbh - DBI database handle
              $_[1] = query - SQL format
   Return:    $a - scalar
              $a = "result"
```

timestamp()

```
Function:    Returns a timestamp
Args:        none
Return:      $_[0] = timestamp: <4-digit YEAR><2-digit MONTH><2-digit DAY>_
               <2-digit HOUR><2-digit MINUTE>
```

get_current_date_mysql()

```
Function:    Returns current date in YYYY-MM-DD mysql format
Args:        none
Return:      $_[0] = date: <4-digit YEAR>-<2-digit MONTH>-<2-digit DAY>
```

timestampsec()

```
Function:    Returns a second-resolution timestamp
Args:        none
Return:      timestamp: <4-digit YEAR><2-digit MONTH><2-digit DAY>_
               <2-digit HOUR><2-digit MINUTE><2-digit SECOND>
```

mysqlimport(file,dbspecs)

```
Function:    load a file into a mysql database using system(mysqlimport)
Return:      $_[0] = records imported
             $_[1] = records deleted
             $_[2] = records skipped
             $_[3] = number of warnings

Args:        $_[0] = filename
             $_[1] = database specs - hashref
```

- db => database name
- user => user name
- pass => password

mysql_runcom(dbh, query, vaues)

```
Function:    Runs an non-SELECT SQL statement
Return:      none
Args:        $_[0] = dbh - DBI database handle
             $_[1] = SQL query command or DBI statement handle
             $_[2] = values- arrayref that hold values for '?' holders
                     in DBI statement handle
```

mysql_createtable(dbh, tablename, spec)

```
Function:    Creates a table (NOTE: if table already exists, drops it first).
Return:      none
Args:        $_[0] = dbh - DBI database handle
             $_[1] = table name
             $_[2] = spec - SQL DDL string

Example uasge:

   pibase::mysql_createtable($dbh,
      $tables->{interface_contacts}->{name},
      $tables->{interface_contacts}->{spec}) ;

   pibase::mysql_runcom($dbh,
      "REPLACE INTO $tables->{interface_contacts}->{meta} ".
      "( bdp_id, table_name) values($bdp_id, ".
      "\"$tables->{interface_contacts}->{name}\")") ;
```

mysql_commandline_query(dbh, query, vaues)

```
Function:    Runs an SQL SELECT statement on the command line,
             optionally performs a sort (on command line), and
             displays the output to a specified file
Return:      none
Args:        $->{db_name} = database name
             $->{sql} = SQL query
             $->{out_fn} = file to display results to
             $->{post_sort} = optional specify field to sort
             $->{sort_order} = optional sort order (defaults ASC)
```

locate_binaries()

```
Function:    Returns location of binaries used in PIBASE associated activities
Return:      $_->{program} = program location.
   perl, zcat, rigor, subset_extractor, altloc_check
Args:        none
```

safe_move()

```
Function:    Safely move a file to a directory (using File::Copy::move),
               retries 14 times, and prints an error if it didnt work
Return:      nothing
Args:        $_[0] = source filename
             $_[1] = target directory
```

safe_copy()

```
Function:    Safely copy a file to a directory (using File::Copy::copy),
               retries 14 times, and prints an error if it didnt work
Return:      nothing
Args:        $_[0] = source filename
             $_[1] = target directory
```

6 pibase::ASTRAL

Perl module to access ASTRAL data

DESCRIPTION

Perl module that contains routines for accessing ASTRAL data, for use in clustering PIBASE interfaces

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

load_asteroids_aln()

```
Title:       load_asteroids_aln()
Function:    Loads an ASTRAL ASTEROIDS alignment file
Args:        $_->{aln_fn}    name of ASTERIODS alignment file
             $_->{seq_fn}    name of corresponding ASTEROIDS sequence file
             $_->{allchains}
             $_->{gdseqh}    data structure holding contents of gdseqh file
             $_->{seqclcont100}
             $_->{seqcl100}
             $_->{doms}      optional hash list of domains to load.
Returns:     parse_aln_raf() alignment structure
```

raf_preload()

```
Title:       raf_preload()
Function:    Loads the ASTRAL raf file
Args:        $_->{fn}        name of the ASTRAL raf file
Returns:     ->{pdb_chain} = "RAF line contents"
```

parse_raf_line()

```
Title:       parse_raf_line()
Function:    Parses a line from ASTRAL raf file
Args:        $_->{line}      RAF line
             $_->{headlength}        length of RAF file header

Returns:     ->{atomresno_first}     ATOM residue number of first residue
             ->{atomresno_last}      ATOM residue number of last residue
             ->{atomresna}   = [ ATOM residue name 1,2, ... ]
             ->{atomresno}   = [ ATOM residue number 1,2, ... ]
             ->{seqresna}    = [ sequence residue name 1,2, ... ]
```

```
->{seqresno}    = [ sequence residue number 1,2, ... ]
->{seqresno2ind}->{seqresno} = index in @{->{seqresno}}
->{ind2seqresno}->{index in @{->{seqresno}}} = seqresno
->{atom2seqresno_back}->{$atomresno} = seqresno
->{atomresno2ind_back}->{$atomresno} = seqresno index
->{seq2atomresno}->{$seqresno} = $atomresno ;
->{atom2seqresno}->{$atomresno} = $seqresno ;
->{atomresno2ind}->{$atomresno} = $#{$res->{seqresno}} ;
->{ind2atomresno}->{$#{$res->{seqresno}}} = $atomresno  ;
```

read_asteroids_aln()

```
Title:       read_asteroids_aln()
Function:    Reads an ASTEROIDS alignment file
Args:        $_->{aln_fn} ASTEROIDS alignment file name
             $_->{seq_fn} corresponding ASTEROIDS sequence file name
             $_->{allchains}->{domain} = pdb chain

Returns:     ->{seq}->{domain} = 'DOMAINSEQVENCE';
             ->{defstring}->{domain} = definition line from alignment
             ->{class}->{domain} = SCOP class
             ->{aln}->{domain} = domain sequence froma alignment
             ->{pdb}->{domain} = PDB code for the domain
             ->{frags}->{domain} = [{b => startresidue, e => endresidue},...]
             ->{alnlength} = alignment length
```

parse_aln_raf()

```
Title:       parse_aln_raf()
Function:    Reads an ASTEROIDS alignment file
Args:        $_->{alndata}  - alignment data from read_asteroids_aln()
             $_->{raf} - RAF data from raf_preload()

Returns:     ->{pos2resno}->{domain}->{alignment position} = ATOM resno
             ->{resno2pos}->{domain}->{ATOM resno} = alignment position
```

load_astral_headers()

```
Title:       load_astral_headers()
Function:    Loads ASTRAL headers
Args:        $_->{fn} - alignment file name
             $_->{raf} - RAF data from raf_preload()

Returns:     ->{gdseqh}->{defstring}->{domain} = domain definition string
             ->{gdseqh}->{class}->{domain} = domain class
             ->{gdseqh}->{pdb}->{domain} = domain PDB code
             ->{gdseqh}->{frags}->{domain} = [{b => startres, e => endres}...]
             ->{gdom}->{domain (w d prefix)} = domain (w g prefix)
```

load_astral_clusters()

```
Title:       load_astral_clusters()
Function:    Loads ASTRAL sequence cluster definitions
Args:        $_->{out} - pointer to hash to hold output
             $_->{pibase_specs} -  pibase_specs structure

Returns:     Nothing - populates the specified $_->{out}
             {out}->{seqcl}->{seq identity}->{scop identifier} = cluster num
             {out}->{seqcl2cont}->{seq identity}->{cluster num} = [scop id,...]
```

get_astral_classlist()

```
Title:       get_astral_classlist()
Function:    Get list of SCOP classes in the ASTRAL compendium
Args:        $_->{pibase_specs} -  pibase_specs structure
Returns:     Nothing - populates the specified $_->{out}
             ->{fam}->{scop_family} = number of domains in the family
             ->{sf}->{scop_superfamily} = number of domains in the superfamily
```

7 pibase::CATH

Interface for CATH domain database data processing

DESCRIPTION

This module contains routines to process and reformat CATH release files for PIBASE import. Files are from: http://www.cathdb.info

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

cath_clean_cddf()

```
Title:       cath_clean_cddf()
Function:    cleans the CATH CDDF file
STDIN:       CATH CDDF file
STDOUT:      cleaned up CATH CDDF file
Args:        nothing
Returns:     nothing
```

cath_parse_cdf_domainlist()

```
Title:       cath_parse_cdf_domainlist()
Function:    cleans the CATH CDDF file
STDIN:       CATH CDDF file
STDOUT:      cleaned up CATH CDDF file
Args:        nothing
Returns:     nothing
```

cath_clean_clf()

```
Title:       cath_clean_clf()
Function:    cleans the CATH CLF file
STDIN:       CATH CLF file
STDOUT:      cleaned up CATH CLF file
Args:        nothing
Returns:     nothing
```

cath_clean_cnf_contraction()

```
Title:       cath_clean_cnf_contraction()
Function:    fixes the concatenation problem with the CATH CNF file
STDIN:       CATH CNF file
```

```
STDOUT:      concat-fixed CATH CNF file
Args:        nothing
Returns:     nothing
```

cath_clean_cnf_contraction()

```
Title:       cath_clean_cnf()
Function:    fixes the concatenation problem with the CATH CNF file
STDIN:       CATH CNF file (preferably concat-fixed)
STDOUT:      cleaned CATH CNF file
Args:        nothing
Returns:     nothing
```

pibase_import_cath_domains()

```
Title:       pibase_import_cath_domains()
Function:    reformats raw CATH tables imported into pibase as generic
               subsets tables
In tables:   1. cath_domain_list
             2. cath_domall_boundaries
             3. cath_names

Out tables:  1. subsets
             2. subsets_class
             3. subsets_details

Args:        nothing
Returns:     nothing
```

8 pibase::PDB

Module to handle PDB functions

DESCRIPTION

Handles general PDB functions: altloc check, filter

FILES

Operates on PDB file

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

altloc_check()

```
Title:       altloc_check()
Function:    checks whether a PDB file for atoms with multiple locations
Args:        $_ = pdb filename
Return:      1 if containts multiple-occurrence atoms, 0 if not
```

altloc_filter()

```
Title:       altloc_filter()
Function:    calls the altloc_filter binary so that each atom occurs once
   Leaves the highest occupied (if occupancy defined) or the first location
   listed in the file.

Args:        $_[0] = source pdb_filename
             $_[1] = output pdb_filename

Returns:     nothing
```

pdb_copy_entry_type()

```
Title:       pdb_copy_entry_type()
Function:    copies PDB pdb_entry_type for pibase import
STDIN:       PDB pdb_entry_type
STDOUT:      pdb_entry_type.pibase_id pibase table
Returns:     nothing
```

pdb_clean_entries_idx()

```
Title:       pdb_clean_entries_idx()
Function:    reformats the PDB entries.idx for pibase import
STDIN:       PDB entries.idx
STDOUT:      pibase.pdb_entries table
Returns:     nothing
```

pdb_clean_obsolete_dat()

```
Title:       pdb_clean_obsolete_dat()
Function:    reformats the PDB obsolete.dat for pibase import
STDIN:       PDB obsolete.dat
STDOUT:      pibase.pdb_obsolete table
Returns:     nothing
```

pdb_clean_release_date()

```
Title:       pdb_clean_release_date()
Function:    reformats the PDB release file for pibase import
STDIN:       NOTDONE PDB obsolete.dat
STDOUT:      pibase.pdb_release table
Returns:     nothing
```

pdb_clean_symop()

```
Title:       pdb_clean_symop()
Function:    Extracts and displays symmetry operators:
STDIN:       PDB symop lines
STDOUT:      pibase.pdb_release table
Returns:     nothing
```

get_pdb_filepath()

```
Title:       get_pdb_filepath()
Function:    returns file path to a PDB entry
Args:        ->{pdb_id} = pdb identifier
             [->{pibase_specs} = $pibase_specs] - optional
Returns:     returns PDB entry filepath
```

nmr_model1_extractor

```
Title:       nmr_model1_extractor()
Function:    extracts first model from NMR PDB files and moves to
             $specs->{pdbnmr_dir}
Args:        none
Returns:     nothing
```

```
Tables in:   pibase.pdb_entries
Files in:    foreach PDB NMR entry: <$specs->{pdb_dir}>/pdb<$pdb_id>.ent
Files out:   foreach PDB NMR entry: <$specs->{pdbnmr_dir}>/<$pdb_id>_1.ent
NOTE: expects PDBs to be stored as pdb<$pdb_id>.ent in one raw directory
```

9 pibase::PDB::chains

Perl module to extract chain listing from a pdb file.

DESCRIPTION

Perl module to parse a pdb file and lists the chains.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

SUB chain_info()

```
Title: chain_info()
Function: calculates chain listing for a pdb file

Args:        $_[0] = pdb_file
             $_[1] = outfile
             $_[2] = bdp_id identifier [optional]
Returns:     nothing

Input file:  PDB file ($_[0])
   http://www.rcsb.org/pdb/docs/format/pdbguide2.2/guide2.2_frame.html
   http://msdlocal.ebi.ac.uk/docs/pdb_format/y_index.html

Output file: Chain listing ($_[1])
   1 pdb identifier (e.g. bdp_id)
   2 Chain number
   3 Chain id
   4 Chain type - p (protein) or n (nucleic acid)
   5 null
   6 null
   7 start residue number
   8 start residue number (integer only)
   9 end residue number
  10 end residue number (integer only)
  11 number of residues
  12 number of atoms
  13 number of het atoms
  14 chain sequence
```

10 pibase::PDB::residues

Perl package to extract residue listing from a PDB file.

DESCRIPTION

Parses a pdb file and lists the residues.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

11 pibase::PDB::sec_strx

Obtain secondary structure assignments from dsspcmbi.

DESCRIPTION

Interface to DSSP to calculate secondary structure for PIBASE structures.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

SUB get_sec_strx()

```
Function: calls DSSP to get a PDB file's residue secondary structure assignment
Args: $_[0] - pdb file
      $_[1] - output file
      $_[2] - bdp identifier - e.g. bdp_id [optional]

Returns:     nothing
Output file:
   1 pdb identifier (e.g. bdp_id)
   2 Chain number
   3 Chain id
   4 residue number
   5 residue number (integer only)
   6 residue name
   7 Polymer type - p (protein) or n (nucleic acid)
```

SUB parse_dssp()

```
Title:    parse_dssp()
Function: parses DSSP output and returns a hash of assignment data
Args: $_[0] - DSSP output file
      $_[1] - output file
      $_[2] - bdp identifier - e.g. bdp_id [optional]

Returns:     $->{detail}->{resno."\n".chain_id} = H|G|I|B|E|T|S|' '
             $->{basic}->{resno."\n".chain_id} = H|B|T|' '
             $->{ordering} = [resno1."\n".chain_id1, resno2."\n".chain_id2...]
             $->{ssnum}->{resno."\n".chain_id} = secondary structure element #
                counts contiguous stretches of same particular detailed sec
                strx assignment)
             $->{ssnum_basic}->{resno."\n".chain_id} = secondary structure
                element number - counts contiguous stretches of basic secondary
                structurea ssignment
```

12 pibase::subsets

Perl module that deals with PDB subsets

DESCRIPTION

Perl module to handle subset operations on PDB files

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

SUB subset_extract()

```
Function: extracts specified residues/chains from PDB file
Args:        $_[0] = PDB file name
             $_[1] = output PDB file name
             $_[2] = chain identifier
             $_[3] = start residue number
             $_[4] = end residue number

Returns:     $_->[i] arrayref of errors

Files IN:    PDB file ($_[0])
Files OUT:   subset PDB file ($_[1])
```

13 pibase::PISA

Perl module of routines that operate on PISA (PDBe's Protein Interfaces, Surfaces, and Assemblies) files.

DESCRIPTION

The PISA.pm module contains routines to process and format PISA release files for PIBASE import.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

pisa_clean_index()

```
Title:       pisa_clean_index()
Function:    cleans PISA index.txt file for pibase import
Args:        none
STDIN:       INDEX
STDOUT:      pibase.pisa_index table
```

get_pisa_filepath()

```
Title:       get_pisa_filepath()
Function:    returns file path to a PISA entry
Args:        ->{pisa_id} = PISA identifier
             [->{pibase_specs} = $pibase_specs] - optional
Returns:     returns PISA entry filepath
```

14 pibase::PQS

Perl module of routines that operate on PQS (EBI's Probable Quaternary Structure server) release files.

DESCRIPTION

The PQS.pm module contains routines to process and format PQS release files for PIBASE import.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

pqs_clean_asalist()

```
Title:       pqs_clean_asalist()
Function:    cleans PQS ASALIST file for pibase import
Args:        none
STDIN:       ASALIST
STDOUT:      pibase.pqs_asalist table
```

pqs_clean_biolist()

```
Title:       pqs_clean_biolist()
Function:    cleans PQS BIOLIST file for pibase import
Args:        none
STDIN:       BIOLIST (preferably pqs_clean_biolist_contract() fixed)
STDOUT:      pibase.pqs_biolist table
```

pqs_clean_biolist_contraction()

```
Title:       pqs_clean_biolist_contraction()
Function:    fixes contraction errors in PQS BIOLIST
Args:        none
STDIN:       BIOLIST
STDOUT:      contraction fixed BIOLIST
```

pqs_clean_list()

```
Title:       pqs_clean_list()
Function:    reformats PQS LIST for pibase import
Args:        none
STDIN:       PQS LIST
STDOUT:      pibase.pqs_list table
```

pqs_clean_ranking()

```
Title:      pqs_clean_ranking()
Function:   reformats PQS RANKING for pibase import
Args:       none
STDIN:      PQS RANKING
STDOUT:     pibase.pqs_ranking table
```

get_pqs_filepath()

```
Title:      get_pqs_filepath()
Function:   returns file path to a PQS entry
Args:       ->{pqs_id} = PQS identifier
            [->{pibase_specs} = $pibase_specs] - optional
Returns:    returns PQS entry filepath
```

15 pibase::SCOP

Package that handles SCOP release files and pibase.

DESCRIPTION

Processes SCOP release files and

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

scop_clean_cla()

```
Title:       scop_clean_cla()
Function:    Processes the SCOP .cla file for pibase import
STDIN:       SCOP CLA file
STDOUT:      pibase.scop_cla table
Args:        none
Returns:     none
```

scop_clean_des()

```
Title:       scop_clean_des()
Function:    Processes the SCOP .des file for pibase import
STDIN:       SCOP DES file
STDOUT:      pibase.scop_des table
Args:        none
Returns:     none
```

scop_clean_hie()

```
Title:       scop_clean_hie()
Function:    Processes the SCOP .hie file for pibase import
STDIN:       SCOP HIE file
STDOUT:      pibase.scop_hie table
Args:        $_->{in_fn} = input file
             $_->{out_fn} = output file
             $_->{header_fl} = flag to generate header in output (default 1)
Returns:     none
```

pibase_import_scop_domains()

```
Title:       pibase_import_scop_domains()
Function:    Processes the SCOP .hie file for pibase import
```

```
Tables in:   pibase.scop_cla
             pibase.scop_des

Tables out:  pibase.subsets
             pibase.subsets_class
             pibase.subsets_details
Args:        none
Returns:     none
```

16 pibase::SGE

Perl module for SGE cluster interaction

DESCRIPTION

Perl module with routines to interact with an SGE cluster, adapted from routines in modtie.pm

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

17 pibase::SUPFAM.pm

DESCRIPTION

This module contains routines to interface with SUPFAM annotation files. Goal is to map SCOP residue numbers onto target sequences using SUPFAM alignments and ASTRAL mapping.

VERSION

fpd091013_0708

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

set_supfam_specs

```
Title:       set_supfam_specs()
Function:    Sets configuration parameters
Args:        None
Returns:     $_->{option} = value; hash of parameters
```

readin_substitution_matrix

```
Title:       readin_substitution_matrix()
Function:    Parses a substitution matrix in matblas format
Args:        ->{matrix_fn} = filename of substitution matrix
             ->{string} = string containing contents of matrix file

Returns:     ->{raw}->{aa1}->{aa2} = substitution matrix score for aa1,aa2
             ->{nl}->{aa1}->{aa2} = normalized aa similarity score
               (see karling_normalize_matrix() for normalization scheme)
```

karlin_normalize_matrix

```
Title:       karlin_normalize_matrix()
Function:    Normalizes a substitution matrix per Karlin and Brocchieri,
               J Bacteriol 1996; and rescaled to range from 0-1:

           0.5 * (1 + ( mat(i,j) / sqrt(|mat(i,i) * mat(j,j)|)))

Args:        ->{matrix}->{aa1}->{aa2} = raw substitution matrix
Returns:     ->{aa1}->{aa2} = normalized similarity score
```

run_pilig_supfam_annotate()

```
Title:       run_pilig_supfam_annotate()
Function:    Maps PIBASE/LIGBASE binding sites onto target sequences
               annotated with SUPERFAMILY domain assignments

Args:        ->{ARGV} = ARGV array reference; parsed to provide:
             ->{ass_fn} = name of SUPERFAMILY domain assignment file
             ->{out_fn} = name of output file
             ->{err_fn} = name of error file
             ->{matrix_fn} = optional substitution matrix, default BLOSUM62
             ->{cluster_fl} = run on an SGE cluster (options in SGE.pm)

Returns:     NOTHING
Displays:     1. seq_id
              2. res_range
              3. classtype
              4. class
              5. bs_type
              6. bs_template
              7. partner_descr
              8. residues
              9. bs_percseqident
             10. bs_percseqsim
             11. bs_numident
             12. bs_numgap
             13. bs_tmpl_numres
             14. bs_fracaln
             15. wholedom_numident
             16. wholedom_aln_length
             17. wholedom_percseqident
             18. wholedom_percseqsim
```

merge_SUPFAM_ASTRAL_alignments

```
Title:       merge_SUPFAM_ASTRAL_alignments()
Function:    Merges SUPERFAMILY alignment string with ASTRAL alignment to
              get SUPERFAMILY annotated target sequence in the ASTRAL
              alignment frame (where it can receive binding site annotations)

Args:        ->{superfam_aln} = SUPERFAMILY alignment string
             ->{astral_aln} = ASTRAL alignment string
             ->{target} = target sequence name
             ->{common} = template sequence name present in both alignments

Returns:     ->{resno2alnpos}->{target resno} = alignment position
             ->{alnpos2resno}->{alignment position} = target resno
```

readin_scop_cla

```
Title:       readin_scop_cla()
Function:    Reads in SCOP cla file (parsing logic from pibase::SCOP)
Args:        ->{fn} = SCOP cla filename
Returns:     ->{px2scopid}->{px_id} = scopid
             ->{scopid2class}->{scopid} = class
             ->{faid2sfid}->{fa_id} = sf_id0
```

get_SUPFAM_selfhit

```
Title:       get_SUPFAM_selfhit()
Function:    Retrieves alignment string from SUPFAM self-hit files for
               a particular SCOP template domain

Args:        ->{supfam_specs} = configuration parameters
             ->{px_id} = SCOP px id
             ->{model_id} = SUPERFAMILY model id

Returns:     self-hit alignment string
```

summarize_results()

```
Title:       summarize_results()
Function:    Parses run_pilig_supfam_annotate() output and reports
               summary of annotations.

Args:        ->{results_fn} = run_pilig_supfam_annotate() output file
Returns:     nothing
Displays:    table describing numbers of proteins,domains,families,residues
              with each kind of annotation
```

18 pibase::aux

Perl interface to auxiliary pibase routines

DESCRIPTION

Perl package that has miscellaneous pibase routines for non-core functions

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

19 pibase::benchmark

Perl package for benchmarking pibase

DESCRIPTION

Contains pibase table structure definitions

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

memusage()

```
Title:       memusage()
Function:    Returns the VmSize of the process
   works on linux /proc/$$/status
```

20 pibase::build

Perl module to build the pibase database

DESCRIPTION

Perl module that executes the PIBASE build protocol.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

21 pibase::calc::interfaces

Perl module to compute protein interfaces.

DESCRIPTION

Perl module that contains routines for computing structural interfaces

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

interface_detect_calc()

```
Title:       interface_detect_calc()
Function:    Detect interfaces in bdp files.
Args:        None
Returns:     Nothing
STDIN:       bdp_id."\t".bdp_path
Files in:    PDB files (as specified in STDIN column 2 (bdp_path))
Files out:
o intersubset_contacts.<hostname>.<timestamp>.<XXXXXX>.<pibase db name>
o patch_residues_tables.meta.<hostname>.<timestamp>.<XXXXXX>.<pibase db name>
o interface_contacts_tables.meta.<hostname>.<timestamp>.<XXXXXX>.
   <pibase db name>
o interface_contacts_special_tables.meta.<hostname>.<timestamp>.<XXXXXX>.
   <pibase db name>

o foreach bdp_id:
   o patch_residues_<bdp_id>.<XXXXXX>.<pibase db name>
   o interface_contacts_<bdp_id>.<XXXXXX>.<pibase db name>
   o interface_contacts_special_<bdp_id>.<XXXXXX>.<pibase db name>
```

_interface_detect_calc__calc_res_pairs().

```
Title:       _interface_detect_calc__calc_res_pairs()
Function:    Calculates residue contacts in a given pdb file.
Args:        $_->{radius} - upper distance limit on inter-atomic contacts calculation [de
             $_->{compress} - compression flag
             $_->{bdp_path} - bdp file path

Return:      $_->{contacts_fn} - kdcontacts output file
             $_->{fields} - kdcontacts file field names
             $_->{field2no}->{field} = i - hash mapping kdcontacts field names to field n
```

```
Files IN:    PDB file ($_->{bdp_path})
Files OUT:   kdcontacts output file ($_->{contacts_fn})
```

cluster_scop_interfaces()

```
Title:       cluster_scop_interfacesj()
Function:    Clusters SCOP-SCOP interfaces using ASTRAL ASTEROIDS alignments.
              and imports to PIBASE if specified
Args:        ->{pibase_specs} = optional
             ->{import_fl} = 1 if to be imported into PIBASE
Returns:     Prints out cluster membership to
              $pibase_specs->{buildfiles}->{scop_interface_clusters}) ;
```

_cluster_interfaces_compare_residue_sets()

```
Title:       _cluster_interfaces_compare_residue_sets()
Function:    Venn comparison of interface residue sets
Args:        ->{aln} = alignment data
             ->{sid1} = domain 1 identifier
             ->{sid2} = domain 2 identifier
             ->{res1}->{ resno1 => 1, resno2 => 1} - residues in set 1
             ->{res2}->{ resno1 => 1, resno2 => 1} - residues in set 2
Returns:     union / (union + diff) of the two sets
```

_cluster_interfaces_load_interface_contacts()

```
Title:       _cluster_interfaces_load_interface_contacts()
Function:    Venn comparison of interface residue sets
Args:        ->{sid1} = domain 1 identifier
             ->{sid2} = domain 2 identifier
Returns:     ->{intres}->{sid1}->{resno1."\n".chain1}= domain 1 res in interface
             ->{intres}->{sid2}->{resno2."\n".chain2}= domain 2 res in interface
             ->{contacts}->{resno1."\n".chain1."\n".resno2."\n".chain2} -
                inter-domain contact
```

_cluster_interfaces_compare_contact_sets()

```
Title:       _cluster_interfaces_compare_contact_ses()
Function:    Venn comparison of interface contact sets
Args:        ->{aln1} = alignment data for domain type 1
             ->{aln2} = alignment data for domain type 2
             ->{sid1} = domain 1 identifier
             ->{sid2} = domain 2 identifier
             ->{cont1} = contacts for interface 1
             ->{cont2} = contacts for interface 2
             ->{revfl}->[i] = reversal flag;
                if 1 switch sid1/2 in ith interface
```

```
Returns:     union / (union + diff) of contacts
```

_cluster_interface_pibase_preload()

```
Title:       _cluster_interface_pibase_preload()
Function:    Preloads PIBASE data necessary for SCOP interface clustering
Args:        ->{astral} = astral data
             ->{aln2} = alignment data for domain type 2
             ->{sid1} = domain 1 identifier
             ->{sid2} = domain 2 identifier
             ->{cont1} = contacts for interface 1
             ->{cont2} = contacts for interface 2
             ->{revfl}->[i] = reversal flag; if 1 switch sid1/2 in ith interface

Returns:     Huge hash of PIBASE data, see code for field explanations
   bdp2contactsfn => $bdp2contactsfn,
   bdp2pdb => $bdp2pdb,
   pdb2bdp => $pdb2bdp,
   bdp_id => $bdp_id,
   sid1 => $sid1,
   sid2 => $sid2,
   class1 => $class1,
   class2 => $class2,
   osid1 => $osid1,
   osid2 => $osid2,
   revfl => $revfl,
   fampairs => $fampairs,
   fampairs_single => $fampairs_single,
   fampairs_single_osid => $fampairs_single_osid,
   pdbchains => $pdbchains,
   chain_2_pdbchain => $chain_2_pdbchain,
   chain_2_start => $chain_2_start,
   chain_2_end => $chain_2_end,
   chain_2_startser => $chain_2_startser,
   chain_2_endser => $chain_2_endser
```

22 pibase::create_raw_table_specs

Module to create perl code with PIBASE db strx

DESCRIPTION

Parses (My)SQL CREATE TABLE statements and displays perl code that defines table structure

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

VERSION

' ;

```
print "fpd".pibase::timestamp()."\n\n" ;
```

print '=head1 DESCRIPTION

The raw_table_specs module contains a hard-coded description of the PIBASE table structures. This file is automatically generated by the pibase::create_raw_table_specs

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

full_table_specs()

```
Title:       full_table_specs()
Args:        none
Returns:     $->{table_name}->{prikey} = primary key field
             $->{table_name}->{field_name}->[i] = name of ith field
             $->{table_name}->{field_spec}->[i] = type of ith field
```

23 pibase::data::access

Perl module of pibase data access routines

DESCRIPTION

Perl package that provides pibase data access routines

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

24 pibase::data::calc

Perl module for pibase data calculation routines

DESCRIPTION

Perl package that provides pibase data calculation routines

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

25 pibase::data::external::ASTRAL

Perl module of ASTRAL data routines

DESCRIPTION

Perl package that provides interface to ASTRAL data files. Includes routine to run MAFFT on domain sequences in the case that the ASTRAL alignments are yet to be released for a new SCOP version.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

26 pibase::data::external::PISA

Perl interface to PISA routines

DESCRIPTION

Perl package that provides interface to PISA data routines

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

27 pibase::data::external::PQS

Perl interface to PQS routines

DESCRIPTION

Perl package that provides interface to PQS data routines

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

28 pibase::interatomic_contacts

Perl module that deals with interatomic contacts

DESCRIPTION

The interatomic_contacts.pm module deals with queries involving interatomic contacts.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

contacts_select()

```
Function:    provides a pseudo-mysql select like interface to [gzipped]
               contacts file in pibase.interatomic_contacts_prototype format
Args:        $fields - arrayref: ['bdp_id', 'resno_1', 'resno_2', 'distance']
Return:      results
```

raw_contacts_select()

```
Title:       raw_contacts_select()
Function:    Allows SQL-like SELECT from the raw interatomic contacts
              tables stored on disk
Args:        $_[0] = source_file
             $_[1] = SQL-like SELECT query
             $_[2] = params
               $_[2]->{maxdist} - distance cutoff
Returns:     $_ - filehandle to of results file
```

contacts_select_inter()

```
Title:       contacts_select_inter()
Function:    Returns inter-domain interatomic-contacts as specified by an
               SQL-like SELECT query from the raw interatomic contacts tables
               stored on disk
Args:        $_[0] = source_file
             $_[1] = SQL-like SELECT query
             $_[2] = resno_2_subset - hash from residue number to domain
               identifier
             $_[3] = params
               $_[3]->{maxdist} - distance cutoff
Returns:     @_ = array of results
```

special_params()

```
Title:       special_params()
Function:    Specifies the parameters (like distance thresholds) for the
               ‘‘special’’ contacts (salt bridges, hydrogen bonds, strong
               hydrogen bonds, disulfide bonds)
Args:        $_[0] = source_file
             $_[1] = SQL-like SELECT query
             $_[2] = resno_2_subset - hash from residue number to domain
               identifier
             $_[3] = params
               $_[3]->{maxdist} - distance cutoff
Returns:     @_ = array of results
```

special_contact()

```
Title:       special_contact()
Function:    given the distance between a pair of atom/residues, decide
               whether it meets dist requirements for a hbond, ssbond,
               or saltbridge
Args:        $_[0] = contacts information
              ->{resna1} = name of residue 1
              ->{atomna1} = name of atom 1
              ->{resna2} = name of residue 2
              ->{atomna2} = name of atom 2
              ->{dist} = distance
             $_[1] = $t - Time::Benchmark timer handle
Returns:     $_ = contact category (none, salt, ssbond, hbond)
```

29 pibase::kdcontacts

Package that parses kdcontacts contacts.

DESCRIPTION

Parses kdcontacts output and displays it in a tab-delimited format ready for import into pibase.interatomic_contacts

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

30 pibase::modeller

Module containing routines to call MODELLER for pibase

DESCRIPTION

Performs MODELLER operations needed by pibase. (still old-school TOP format)

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

subsets_2_modpick()

```
Function:    Converts subsets_details to MODELLER SELECTION SEGMENTS
Args:        $_[0] = subset_id
             $_[1] = DBI db handle to pibase
Return:      $_->[] arrayref of MODELLER pick statements to select domain
```

subsetdef_2_mod_pick()

```
Function:    Converts domain definition to MODELLER pick statements
Args:        $_[0] = arrayref of chain_id
             $_[1] = arrayref of start_resno
             $_[2] = arrayref of end_resno
Return:      $_->[] arrayref of MODELLER pick statements to select domain
```

get_salign (modeller_bin, bdp_file)

```
Title:       get_salign()
Function:    Calls MODELLER.SALIGN to structurally align two pdb files
Args:        $_->{pdb_fn_1} - name of pdb file 1
             $_->{pdb_fn_2} - name of pdb file 2
             $_->{modeller_bin} - name of MODELLER binary file
```

get_salign_seqseq (modeller_bin, bdp_file)

```
Title:       get_salign_seqseq()
Function:    Calls MODELLER.SALIGN to sequence align two pdb files
Args:        $_->{pdb_fn_1} - name of pdb file 1
             $_->{pdb_fn_2} - name of pdb file 2
             $_->{modeller_bin} - name of MODELLER binary file
```

OLD_modeller_subset_sasa (modeller_bin, bdp_file, picks)

```
Title:       OLD_modeller_subset_sasa()
Function:    Calculate the solvent accessible surface area of a pdb file
Args:        $_[0] = modeller_binary location
             $_[1] = pdb file location
             $_[2] = pick statements for domain
Returns:     $_[0] = SASA of domain (get_sasa() data structure)
             $_[1] = error_fl - error flag
```

Specify the temporary alignment TOP file, and the output alignment file.
Generate the actual TOP file.
Specify the location of the MODELLER LOG file.
Run the TOP file through MODELLER.

get_sasa(modeller_bin, bdp_file, picks)

```
Title:       get_sasa()
Function:    parse modeller sasa (psa)
Args:        $_->{surftyp} = type of MODELLER surface area
             $_->{pdb_fn} = pdb file location
             $_->{modeller_bin} = modeller binary file
Returns:     $_[0] = results
             $_[1] = resno_rev
             $_[2] = sasa
             $_[3] = error_fl
```

calc_sasa()

```
Title:       calc_sasa()
Function:    Run and parse modeller sasa (psa)
Args:        $_->{surftyp} = type of MODELLER surface area
             $_->{pdb_fn} = pdb file location
             $_->{modeller_bin} = modeller binary file
Returns:     ->{res_sasa}->{all_sum|mc_sum|sc_sum|p_sum|nonp_sum}->[i] =
                 SASA information for residue record i
             ->{sasa_resno_rev}->{"residuenumber_chain"} = record number
             ->{full_sasa}->{p|nonp|mc|sc|all} = totals of residue sasa records
             ->{error_fl} => $error_fl,
             ->{atm_sasa}->{p|nonp|mc|sc|all} = totals of ATOM sasa records
```

get_dihedrals()

```
Title:       get_dihedrals()
Function:    run and parse modeller dihedrals (dih)
Args:        $_[0] = bdp_file
             $_[1] = modeller_bin
```

```
Returns:     $_[0] = results
             $_[1] = resno_rev
             $_[2] = error_fl
```

get_vol()

```
Function:    Run and parse MODELLER volume (psa)
Args:        $_[0] = bdp_file
             $_[1] = MODELLER binary file
Returns:     $_[0] = results
             $_[1] = resno_rev
             $_[2] = sasa
             $_[3] = error_fl
```

cutpdb()

```
Title:       cutpdb()
Function:    Uses MODELLER to extrct domain from a pdb file
Args:        $_[0] = MODELLER binary location
             $_[1] = pdb file location
             $_[2] = MODELLER pick statements
             $_[3] = output pdb file name

Returns:     $_[0] =
             $_[1] = resno_rev
             $_[2] = error_fl

FILE in:     pdb file ($_[1])
FILE out:    domain pdb file ($_[3])
```

parse_ali()

```
Title:       parse_ali()
Function:    reads in a modeller PIR format alignment and returns residue
             number equivalence hashes
Args:        $_->{ali_fn} alignment file
             $_->{modpipe_newstyle_orderswitch}
               - 1 (Default) if new style MODPIPE run
               - reordered sequences in the alignment file
Results:     ->{seq} = $seq ;
             ->{resno_start} = $resno_start ;
             ->{resno_end} = $resno_end ;
             ->{chain_start} = $chain_start ;
             ->{chain_end} = $chain_end ;
             ->{alipos_2_serial} = $alipos_2_serresno ;
             ->{alipos_2_chainno} = $alipos_2_chainno ;
             ->{alipos_2_resna} = $alipos_2_resna ;
```

```
                ->{maxlength} = $maxlength;
```

get_resequiv()

```
   Title:       get_resequiv()
   Function:    Determines a mapping between residues in two pdb files.
                  Returns a combine serial residue number/positioning from
                  get_resequiv_serial with residue_info()
   Args:        ->{modeller_bin} = MODELLER binary location
                ->{pdb_fn_1} = PDB file 1 location
                ->{pdb_fn_2} = PDB file 2 location
   Returns:     $->[0]->{resno1} = resno2. maps from resno1 in first pdb file
                  to the aligned residue in the second pdb file
                $->[1]->{resno2} = resno1. maps from resno2 in second pdb file
                  to the aligned residue in the first pdb file
```

get_resequiv_serial()

```
   Title:       get_resequiv_serial()
   Function:    Determines residue equivalencies between two pdb files by
                  aligning them structurally using MODELLER.SALIGN
   Args:        ->{modeller_bin} = MODELLER binary location
                ->{pdb_fn_1} = location of pdb file name 1
                ->{pdb_fn_2} = location of pdb file name 2
   Returns:     parse_ali() alignment structure
```

31 pibase::pilig

Perl module for pibase-ligbase overlap calculations

DESCRIPTION

Perl module with routines to cross-query pibase and ligbase to get small molecule - protein interaction site overlap statistics

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

32 pibase::pilig

Perl module for pibase-ligbase overlap calculations

DESCRIPTION

Perl module with routines to cross-query pibase and ligbase to get small molecule - protein interaction site overlap statistics

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

33 pibase::pilig

Perl module for pibase-ligbase overlap calculations

DESCRIPTION

Perl module with routines to cross-query pibase and ligbase to get small molecule - protein interaction site overlap statistics

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

34 pibase::raw_table_specs- module that specifies pibase table structures

VERSION

fpd100910_1640

DESCRIPTION

The raw_table_specs module contains a hard-coded description of the PIBASE table structures. This file is automatically generated by the pibase::create_raw_table_specs

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

full_table_specs()

```
Title:      full_table_specs()
Args:       none
Returns:    $->{table_name}->{prikey} = primary key field
            $->{table_name}->{field_name}->[i] = name of ith field
            $->{table_name}->{field_spec}->[i] = type of ith field
```

35 pibase::residue_math

Package that handles residue number operations

DESCRIPTION

The pibase::resno module performs common operations on residue numbers.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

residue_int(resno)

```
Title:       residue_int()
Function:    Seperates the integer and insertion code of a residue number
   NOTE: assumes insertion code is always alphanumeric
Args:        residue number (5 character - number and insertion code)
Returns:     $_[0] integer portion of the residue number
             $_[1] insertion code of the residue number
```

residue_add(residue number, increment)

```
Title:       residue_add()
Function:    Adds an integer increment to a residue number
Args:        $_[0] = residue number (full)
             $_[1] = increment
Returns:     $_ = new residue number
```

residue_comparison(resno1, resno2)

```
Title:       residue_comparison(resno1, resno2)
Function:    Compares 2 residues to determine which one is greater.
Args:        $_[0] = residue number 1
             $_[2] = residue number 2
Returns:     $_ = comparison result: 0 = equal,
                                     1 = first is greater,
                                     2 = second is greater.
```

residue_inrange(resno, start, end)

```
Title:       residue_inrange(resno, start, end)
Function:    Checks if a residue number is within a range of residue numbers.
Args:        $_[0] = residue number
             $_[1] = start residue number range
```

```
                $_[1] = end residue number range
Returns:        Comparison result: 0 = out of range, 1 = in range.
```

36 pibase::specs

Perl package containing definition of pibase table structures

DESCRIPTION

Contains pibase table structure definitions

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

table_spec(@tablelist)

```
Title:       table_spec()
Function:    Return mysql DDL format table specs.
Returns:     $_ hashref pointing from tablename to specs
               $_->{i} = specs for ith table
```

SUB sql_table_spec(@tablelist)

```
Title:       sql_table_spec()
Function:    Return mysql DDL format table specs.
Args:        $_ hashref pointing from tablename to specs
             $_->{i} = specs for ith table
```

37 pibase::tables_on_disk

Perl module that provides an SQL like query interface to tables stored on disk.

DESCRIPTION

Perl module to interface with tables stored on disk.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

select_tod()

```
Title:      select_tod()
Function:   provides a pseudo-mysql select like interface to [gzipped]
            contacts file in pibase.interatomic_contacts_prototype format

Args:       $_[0]- source file
            $_[1]- arrayref ['bdp_id', 'resno_1', 'resno_2', 'distance']
            $_[2]- table name
            $_[3]- where clause

Returns:    @_ - array of arrays
            $_[i][j] = jth field of ith result
```

38 pibase::web

Collection of routines for PIBASE web interface

DESCRIPTION

This module provides routines for the PIBASE web interface.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

greeting()

```
Title:      greeting()
Function:   Provides pibase web site greeting
Args:       $_->{base} [optional] = website base url
Return:     html code for greeting section of pibase webpage
```

closing()

```
Title:      closing()
Function:   Provides pibase web site closing
Args:       $_->{base} [optional] = website base url
Return:     html code for closing section of pibase webpage
```

find_object()

```
Title:      find_object()
Function:   Routine for finding complex, interface, or domain
Args:       $_->{base} [optional] = website base url
Return:     STDOUT html page of search results
```

display_results()

```
Title:      display_results()
Function:   Formats a list of query results (find_object()) for html viewing
Args:       $_->{input}->{object_type} = complexes|interfaces|domains
            $_->{details_link}->{object_type} = construct for detail URL link
            $_->{view_link}->{object_type} = construct for viewer URL link
            $_->{results_field}->[i] = ith result field header
            $_->{data}->[i]->[j] = jth field of ith record
            $_->{cgi_h} = CGI handle
            $_->{base} [optional] = website base url
            $_->{basecgi} [optional] = website base url
```

```
Returns:     STDOUT html page of search results
```

get_object_details()

```
Title:       get_object_details()
Function:    Formats a list of query results (find_object()) for html viewing
Args:        $_->{input}->{object_type} = complexes|interfaces|domains
             $_->{details_link}->{object_type} = construct for detail URL link
             $_->{view_link}->{object_type} = construct for viewer URL link
             $_->{results_field}->[i] = ith result field header
             $_->{data}->[i]->[j] = jth field of ith record
             $_->{cgi_h} = CGI handle
             $_->{base} [optional] = website base url
             $_->{basecgi} [optional] = website base url
Returns:     STDOUT html page of search results
```

foldtext()

```
Title:       foldtext()
Function:    Wrapping code for a string with specified folding characters
Args:        $_->{string} = 'ORIGINALSTRING'
             $_->{wrap} = wrapping length
             $_->{foldchar} = folding charcter [optional - defaults to newline]
Returns:     $_ = "ORIG\nINAL\nSTRI\nNG" ;
```

get_color_codes()

```
Title:       get_color_codes()
Function:    Returns rgb decimal and hex values for a specified color name
Args:        $_->{name} = color name
Returns:     $_->{rgb} = "R G B" (0-1 scale)
             $_->{rgb255} => "R,G,B" (0-255 scale)
             $_->{hexcode} => "RRGGBB" hex code
```

pngconvert_bdp_interaction_topology_graph()

```
Title:       pngconvert_bdp_interaction_topology_graph()
Function:    Iterates over all eps format topology graphs and converts to
              PNG using ImageMagick
Args:        $_->{pibase_specs} = pibase_specs [optional - if not get_specs()]
Returns:     Nothing
```

view_object()

```
Title:       view_object()
Function:    Retrieves and displays to STDOUT PIBASE structure file for
              domain, interface, or complex
```

```
Args:        $_->{subset_id} = subset_id of domain
             $_->{subset_id_1} = subset_id of domain 1 of an interface
             $_->{subset_id_2} = subset_id of domain 2 of an interface
             $_->{bdp_id} = bdp_id of a complex
             $_->{subset_source_id} = domain classification system of a complex
Returns:     STDOUT html page of search results
```

view_interface_subset2rasmol()

```
Title:       view_object()
Function:    Returns rasmol script commands to define a series of domains
Args:        $_->{dbh} = PIBASE database handle
             $_->{subsets} = [subset_id_1, 2, ...]
             $_->{colors} = [name of color to use for domain 1, 2, ...]
Returns:     STDOUT prints rasmol script commands
```

39 pibase::web_pilig

Collection of routines for PIBASE.ligands web interface

DESCRIPTION

This module provides routines for the PIBASE.ligands web interface.

AUTHOR

Fred P. Davis, HHMI-JFRC (davisf@janelia.hhmi.org)

SUBROUTINES

greeting()

```
Title:      greeting()
Function:   Provides pibase web site greeting
Args:       $_->{base} [optional] = website base url
Return:     html code for greeting section of pibase webpage
```

closing()

```
Title:      closing()
Function:   Provides pibase web site closing
Args:       $_->{base} [optional] = website base url
Return:     html code for closing section of pibase webpage
```

find_object()

```
Title:      find_object()
Function:   Routine for finding complex, interface, or domain
Args:       $_->{base} [optional] = website base url
Return:     STDOUT html page of search results
```

display_results()

```
Title:      display_results()
Function:   Formats a list of query results (find_object()) for html viewing
Args:       $_->{input}->{object_type} = complexes|interfaces|domains
            $_->{details_link}->{object_type} = construct for detail URL link
            $_->{view_link}->{object_type} = construct for viewer URL link
            $_->{results_field}->[i] = ith result field header
            $_->{data}->[i]->[j] = jth field of ith record
            $_->{cgi_h} = CGI handle
            $_->{base} [optional] = website base url
            $_->{basecgi} [optional] = website base url
```

```
Returns:     STDOUT html page of search results
```

get_object_details()

```
Title:       get_object_details()
Function:    Formats a list of query results (find_object()) for html viewing
Args:        $_->{input}->{object_type} = complexes|interfaces|domains
             $_->{details_link}->{object_type} = construct for detail URL link
             $_->{view_link}->{object_type} = construct for viewer URL link
             $_->{results_field}->[i] = ith result field header
             $_->{data}->[i]->[j] = jth field of ith record
             $_->{cgi_h} = CGI handle
             $_->{base} [optional] = website base url
             $_->{basecgi} [optional] = website base url
Returns:     STDOUT html page of search results
```

foldtext()

```
Title:       foldtext()
Function:    Wrapping code for a string with specified folding characters
Args:        $_->{string} = 'ORIGINALSTRING'
             $_->{wrap} = wrapping length
             $_->{foldchar} = folding charcter [optional - defaults to newline]
Returns:     $_ = "ORIG\nINAL\nSTRI\nNG" ;
```

get_color_codes()

```
Title:       get_color_codes()
Function:    Returns rgb decimal and hex values for a specified color name
Args:        $_->{name} = color name
Returns:     $_->{rgb} = "R G B" (0-1 scale)
             $_->{rgb255} => "R,G,B" (0-255 scale)
             $_->{hexcode} => "RRGGBB" hex code
```

pngconvert_bdp_interaction_topology_graph()

```
Title:       pngconvert_bdp_interaction_topology_graph()
Function:    Iterates over all eps format topology graphs and converts to
              PNG using ImageMagick
Args:        $_->{pibase_specs} = pibase_specs [optional - if not get_specs()]
Returns:     Nothing
```

view_object()

```
Title:       view_object()
Function:    Retrieves and displays to STDOUT PIBASE structure file for
              domain, interface, or complex
```

```
Args:        $_->{subset_id} = subset_id of domain
             $_->{subset_id_1} = subset_id of domain 1 of an interface
             $_->{subset_id_2} = subset_id of domain 2 of an interface
             $_->{bdp_id} = bdp_id of a complex
             $_->{subset_source_id} = domain classification system of a complex
Returns:     STDOUT html page of search results
```

view_interface_subset2rasmol()

```
Title:       view_object()
Function:    Returns rasmol script commands to define a series of domains
Args:        $_->{dbh} = PIBASE database handle
             $_->{subsets} = [subset_id_1, 2, ...]
             $_->{colors} = [name of color to use for domain 1, 2, ...]
Returns:     STDOUT prints rasmol script commands
```