turboOrtho

0.9.7.6

Generated by Doxygen 1.7.4

Sat Dec 31 2011 17:51:08

Contents

Chapter 1

Todo List

Class pipe_bucket Should be considered removed, replaced by preestimated blocks used by each of the consquative threads

Class pipe_norm If the norm_t∗∗ object is already up to data, this step is a waste of time. Consider using an altnerative approach.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

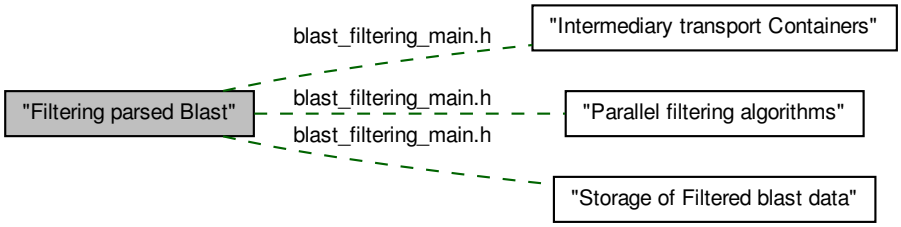
Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 ”Filtering parsed Blast”

Collaboration diagram for "Filtering parsed Blast":



Classes

- class blast_filtering

  Produces a filtered output of the input.

Files

- file blast_filtering_main.h

  The launcher of the filtering library.

Typedefs

- typedef class blast_filtering blast_filtering_t

  Produces a filtered output of the input.

5.1.1 Detailed Description

Executing code for filtering

5.1.2 Typedef Documentation

5.1.2.1 typedef class blast_filtering blast_filtering_t

Produces a filtered output of the input.

Author

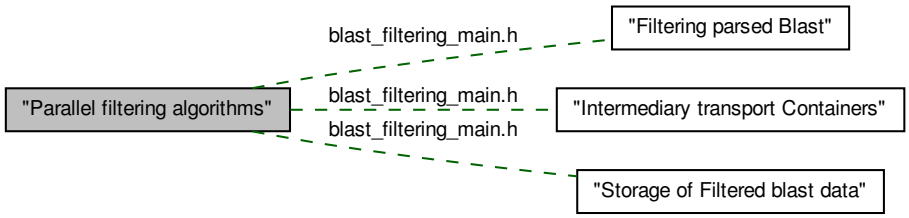
Ole Kristian Ekseth (oekseth)

Date

31.12.2011 by oekseth (clean-up)

5.2 "Parallel filtering algorithms"

Collaboration diagram for "Parallel filtering algorithms":



Classes

- class pipe_binary

  Filters orthologs- and inparalogs in parallel.
- class pipe_bucket

Produces buckets of numbered items to parse.

- class pipe_merge

  Merges containers building a filtered set of orthologs- and inparalogs.

- class pipe_norm

  *Updates an norm_t** object, if it's not done so in a previous phase.*

- class pipe_struct

  Either builds co-orthologs or builds the strings for the result file.

- class pipe_write

  Builds (writes) the result files, consisting of the strings given.

Files

- file blast_filtering_main.h

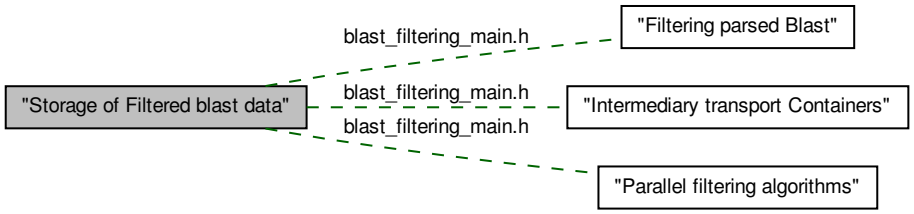
  The launcher of the filtering library.

5.2.1 Detailed Description

Parallel blocks used during the filtering- and output process.

5.3 "Storage of Filtered blast data"

Collaboration diagram for "Storage of Filtered blast data":



Classes

- class build_string

  Produces a row of chars for the protein given.

Files

- file blast_filtering_main.h

  The launcher of the filtering library.

Typedefs

- typedef build_string build_string_t

  Produces a row of chars for the protein given.
- typedef enum write_list write_list_t

  In order to identify the files.

Enumerations

- enum write_list { **blast_ortho_pairs_list_numbers**, **blast_inparalog_list_numbers**, **blast_complete_list_numbers**, **blast_ortho_relations_list_numbers** }

  In order to identify the files.

Variables

- static const uint size_write_list_t = 4

  The number of elements in the enum write_list.

5.3.1 Detailed Description

Data container used during the filter- and output process.

5.3.2 Typedef Documentation

5.3.2.1 typedef build_string build_string_t

Produces a row of chars for the protein given.

Author

Ole Kristian Ekseth (oekseth

Date

16.09.2011 by oekseth (asserts)
31.12.2011 by oekseth (cleanup)

5.3.2.2 typedef enum write_list write_list_t

In order to identify the files.

Author

Ole Kristian Ekseth (oekseth

5.3.3 Enumeration Type Documentation

5.3.3.1 enum write_list

In order to identify the files.

;

Author

Ole Kristian Ekseth (oekseth

5.3.4 Variable Documentation

5.3.4.1 const uint size_write_list_t = 4 [static]

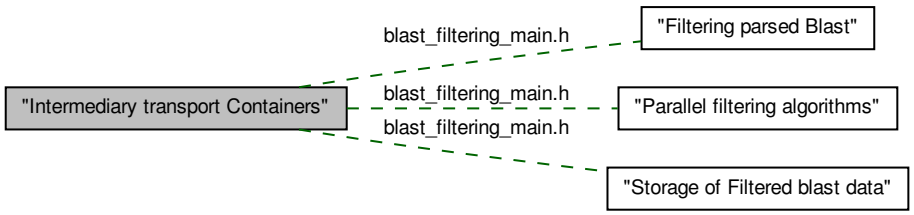
The number of elements in the enum write_list.

Author

Ole Kristian Ekseth (oekseth

5.4 "Intermediary transport Containers"

Collaboration diagram for "Intermediary transport Containers":



Classes

- struct bucket_pipe_binary
  A transport container.

Files

- file blast_filtering_main.h
  The launcher of the filtering library.

Typedefs

- typedef struct bucket_pipe_binary bucket_pipe_binary_t
  A transport container.

5.4.1 Detailed Description

Data container stored intermediary as transported between two pipes.

5.4.2 Typedef Documentation

5.4.2.1 typedef struct bucket_pipe_binary bucket_pipe_binary_t

A transport container.

Author

Ole Kristian Ekseth (oekseth)

Date

25.12.2011 by oekseth (cleanup).

5.5 "Algorithms for serial Filtering"

Classes

- class taxon_pair
  Defines the next protein to work on.

Typedefs

- typedef class taxon_pair taxon_pair_t
  Defines the next protein to work on.

5.5.1 Detailed Description

Data container used for specific filtering operations.

5.5.2 Typedef Documentation

5.5.2.1 typedef class taxon_pair taxon_pair_t

Defines the next protein to work on.

Author

Ole Kristian Ekseth (oekseth)

Date

24.12.2011 by oekseth (removed calls to 'extern' variables to ease the inclusion of this class as a libary)
31.12.2011 (cleanup)

Chapter 6

Class Documentation

6.1 blast_filtering Class Reference

Produces a filtered output of the input.

`#include <blast_filtering.h>`

Public Member Functions

- void print_class_info ()

  Prints information describing this class.
- void init_values (cmd_list ∗cmd)

  Connects internal values to the object given as param, enabling the terminal updating internal vars.
- void set_values (bool _DEBUG_NORM, bool _PRINT_NORMALIXATION_BASIS, bool _USE_EVERYREL_AS_ARRNORM_BASIS, char ∗_FILE_BINARY_LOCATION, char _SEPERATOR, int _CPU_TOT)

  Initializes values after those set in the blast parsing (dound in library blast_parsing):
- void start_filtering (log_builder_t ∗log, bp_container_t bp)

  Executes the main operation for the filtering:
- void free_memory ()

  De-allocates memory for this object.
- blast_filtering (cmd_list ∗cmd)

  The constructor.

Static Public Member Functions

- static cmd_list ∗ init_cmd_list (char ∗DEFAULT_OPTION_NAME, uint &DEFAULT_-OPTION_NAME_COUNT, char ∗FILE_INPUT_NAME)

  Initiates the list for parsing input arguments fromthe terminal:

- static void build_cmd_list (cmd_list *cmd, int argc, char *argv[])

  Maps the internal variables to the input given from the terminal.
- static void close (blast_filtering *&obj)

  De-allocates memory for the object given as input.
- static void assert_class (bool print_info)

  The assert method:

6.1.1 Detailed Description

Produces a filtered output of the input.

Remarks

The purpose of this wrapper module is handling a the filtering process given an input of a taxa_t- and list_file_parse containers, producing a filtered output.

Author

Ole Kristian Ekseth (oekseth)

Date

21.12.2010 by Ole Kristian Ekseth (init)
18.08.2011 by Ole Kristian Ekseth (Cleaning.)
24.12.2011 by oekseth (removed calls to 'extern' variables to ease the inclusion of thisclass as a libary)
31.12.2011 by oekseth (clean-up)

6.1.2 Member Function Documentation

6.1.2.1 void blast_filtering::start_filtering (log_builder_t * *log*, bp_container_t *bp*)

Executes the main operation for the filtering:

Parameters

<*log*>	The log object to store measurements in.
<*bp*>	The object containing the basis whom the filtering will work on.

The documentation for this class was generated from the following file:

- blast_filtering.h

6.2 bucket_norm Struct Reference

Data container holding changes in the normative array.

`#include <bucket_norm.h>`

Public Member Functions

- void free_mem (const uint taxon_length)

  Deallocates the memory.
- bucket_norm (norm_t **_arrNorm)

  Constructs the class given the input params.

Static Public Member Functions

- static bucket_norm * init (norm_t **_arrNorm)

  Initializes the class.
- static void assert_class (const bool print_info)

  Asserts the class given with test functions.

Public Attributes

- norm_t ** arrNorm

  Holds the basis for the normalization values.

6.2.1 Detailed Description

Data container holding changes in the normative array.

Author

Ole Kristian Ekseth (oekseth)

Date

02.11.2011 by oekseth (initial)

The documentation for this struct was generated from the following file:

- bucket_norm.h

6.3 bucket_pipe_binary Struct Reference

A transport container.

`#include <bucket_pipe_binary.h>`

Public Member Functions

- bool isNotEmpty ()
- void free_structdata ()

  De-allocates memory for the list_file_struct object.
- void free_arrNorm (const uint taxon_length)

  Deallocates memory for the ∗∗norm object given the length of it.
- void free_mem (const uint taxon_length)

  Deallocates the memory for this object.
- bucket_pipe_binary ()

  The constructor.
- bucket_pipe_binary (norm_t ∗∗_arrNorm, list_file_struct_t ∗_structData)

  The constructor.

Public Attributes

- norm_t ∗∗ arrNorm
- list_file_struct_t ∗ structData

  Holds the filtered blast data.

6.3.1 Detailed Description

A transport container.

Author

Ole Kristian Ekseth (oekseth)

Date

21.12.2010 by oekseth (init).
25.12.2011 by oekseth (cleanup).

6.3.2 Member Function Documentation

6.3.2.1 bool bucket_pipe_binary::isNotEmpty ( )

Returns

true if data is set

6.3.3 Member Data Documentation

6.3.3.1 norm_t∗∗ bucket_pipe_binary::arrNorm

Returns

the ∗∗norm object holding the basis for the normalization procedure.

The documentation for this struct was generated from the following file:

- bucket_pipe_binary.h

6.4 build_string Class Reference

Produces a row of chars for the protein given.

`#include <build_string.h>`

Public Member Functions

- void write_data_to_file (FILE *out_file, const bool PIPE)

  Writes the string to the file, and clears the memory.
- void setHeader (uint world_index_in)

  Sets the header for the string to be written to the output file.
- void avoidPrinting ()

  Called when no data shall be printed to the stream.
- void writeRow (bool _has_data)

  If set to true, the data in the poiners are written to a file.
- bool hasData ()
- char * begin ()
- void append (char *first, char *last)

  Appends the chars to the buffer.
- void copy (char *start, char *end)

  adds the content of the input to the string located in this object.
- void copy_struct (build_string row)

  adds the content of the input to the string located in this object.
- void end_blast_line ()

  Appends the line-end characters specified for the 'blast' file format.
- long int getIndex ()

  Returns the size of the index.
- void finalize ()

  Appends the line end ('\0' character)
- uint size ()

  Length of sequence.
- mem_loc resize_if_to_large (mem_loc length_argument)

  Increases the size of the buffer if it does not fit.
- void free_mem ()

  Frees the memory allocated for this object.
- build_string (uint _size_blast_all)

  The constructor.

Static Public Member Functions

- static build_string * allocate_class (uint size)

  Allocates a list of 'this' class, but do not initialize it.

6.4.1 Detailed Description

Produces a row of chars for the protein given.

Author

Ole Kristian Ekseth (oekseth

Date

21.12.2010 by oekseth (initial)
16.09.2011 by oekseth (asserts)
31.12.2011 by oekseth (cleanup)

6.4.2 Member Function Documentation

6.4.2.1 char* build_string::begin ( )

Returns

pointer to beginning of the sequence the string represents.

6.4.2.2 void build_string::copy (char * *start*, char * *end*)

adds the content of the input to the string located in this object.

Parameters

<*start*>	Pointer to the first char in the sequence of chars.
<*end*>	Pointer to the last char in the sequence of chars.

6.4.2.3 bool build_string::hasData ( )

Returns

true if data shall be printed

6.4.2.4 mem_loc build_string::resize_if_to_large (mem_loc *length_argument*)

Increases the size of the buffer if it does not fit.

Parameters

<length_argument>	the length of the input.

Returns

the size of the argument to be copied.

Remarks

To be used at insertion of a new part of the char array

The documentation for this class was generated from the following file:

- build_string.h

6.5 pipe_binary Class Reference

Filters orthologs- and inparalogs in parallel.

`#include <pipe_binary.h>`

Public Member Functions

- struct taxon_pair * init_taxon_pair (uint taxon_start, uint taxon_length, uint __taxon_length, int n_threads)

  Builds the set of blocks to be used during the parsing:
- void free_data ()

  De-allocates the data bounded by this class.
- void free_additional_blocks ()

  De-allocates both the internal temporary objects of type list_file_struct for all of the threads, in addition to de-allocating the memory reserved for this object.
- void * operator() (void *item)

  The method of parallisation.
- pipe_binary (uint _taxon_length, const bool _inparalog_operation, const bool _use_everyrel_as_arrnorm_basis, const uint _n_threads, log_builder_t *_log, id_simil_list &_listOrtho, list_file_parse_t *_listParseData, short int _AMINO_LIMIT, float _max_input_value, float _MIN_SIMILARITY_LIMIT, bool _use_improved_overlap_algo, bool _DEBUG_PRINT_DISCARDED_PAIRS, bool _PRINT_OVERLAP_VALUES_ABOVE, bool _PRINT_NORMALIXATION_BASIS, bool _DEBUG_NORM, taxa_t *listTaxa, bool _MODE_PAIRWISE_OUTPUT_ABC, bool MODE_PAIRWISE_OUTPUT_MCL, char *FILE_BINARY_LOCATION)

  The constructor.

Static Public Member Functions

- static void assert_class (const bool print_info)

  The main test function for this class.

Public Attributes

- list_file_struct ** l_fileStruct

  The object containing list_file_struct during the building process for each thread id.

6.5.1 Detailed Description

Filters orthologs- and inparalogs in parallel.

Author

Ole Kristian Ekseth (oekseth)

Date

18.03.2011 by oekseth (initial)
15.09.2011 by oekseth (asserts)
24.12.2011 by oekseth (removed calls to 'extern' variables to ease the inclusion of thisclass as a libary)
25.12.2011 by oekseth (cleanup).

The documentation for this class was generated from the following file:

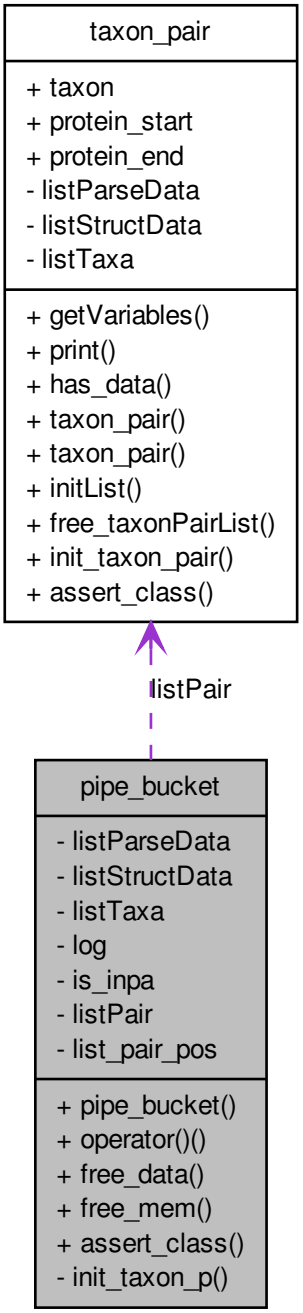
- pipe_binary.h

6.6 pipe_bucket Class Reference

Produces buckets of numbered items to parse.

`#include <pipe_bucket.h>`

Collaboration diagram for pipe_bucket:



Public Member Functions

- pipe_bucket (const bool inpa_ops, uint taxon_start, uint taxon_end, uint taxon_-length, log_builder_t *_log, list_file_parse_t *&_listParseData, list_file_struct_t *&_listStructData, taxa_t *_listTaxa)

  The constructor.
- void * operator() (void *item)

  The method of parallisation.
- void free_data ()

  De-allocates the memory of this object.
- void free_mem ()

  De-allocates the memory of this object.

Static Public Member Functions

- static void assert_class (const bool print_info)

  The main test function for this class.

6.6.1 Detailed Description

Produces buckets of numbered items to parse.

Goal is to make the threads in the rest of the pipe effective:

Todo

Should be considered removed, replaced by preestimated blocks used by each of the consquative threads

Author

Ole Kristian Ekseth (oekseth)

Date

21.12.2010 by oekseth (initial)
16.09.2011 by oekseth (asserts)
24.12.2011 by oekseth (removed calls to 'extern' variables to ease the inclusion of thisclass as a libary)

The documentation for this class was generated from the following file:

- pipe_bucket.h

6.7 pipe_merge Class Reference

Merges containers building a filtered set of orthologs- and inparalogs.

```
#include <pipe_merge.h>
```

Public Member Functions

- void set_arrNorm (norm_t **norm)

  *Sets the **norm object with the param given.*
- norm_t ** get_arrNorm ()
- void setFileStruct (list_file_struct *arg)

  *Sets the *list_file_struct object with the param given.*
- list_file_struct * getFileStruct ()
- void * operator() (void *item)

  The method of parallisation.
- pipe_merge (uint _taxon_length, const bool _use_everyrel_as_arrnorm_basis, const pipe_t type, log_builder_t *_log, list_file_struct_t *&_listStructData, taxa_t *_listTaxa)

  The constructor.

6.7.1 Detailed Description

Merges containers building a filtered set of orthologs- and inparalogs.

Remarks

Inputs an object of type 'bucket_pipe_binary', containing blocks of information to be merged.

- Merges containers of type list_file_struct and type **norm.

Author

Ole Kristian Ekseth (oekseth)

Date

21.12.2010 by oekseth (initial)
16.09.2011 by oekseth (asserts)
24.12.2011 by oekseth (removed calls to 'extern' variables to ease the inclusion of thisclass as a libary)
25.12.2011 by oekseth (cleanup).

6.7.2 Member Function Documentation

6.7.2.1 norm_t** pipe_merge::get_arrNorm ( ) [inline]

Returns

the **norm object

6.7.2.2 list_file_struct* pipe_merge::getFileStruct ( ) [inline]

Returns

the processed *list_file_struct object.

The documentation for this class was generated from the following file:

- pipe_merge.h

6.8 pipe_norm Class Reference

Updates an norm_t** object, if it's not done so in a previous phase.

`#include <pipe_norm.h>`

Public Member Functions

- void * operator() (void *item)

  The method of parallisation.
- pipe_norm (uint _taxon_length, const bool _use_everyrel_as_arrnorm_basis, const pipe_t type, log_builder_t *_log)

  The constructor:

Public Attributes

- norm_t ** arrNorm

  The object containing the basis for the normalization procedure.

6.8.1 Detailed Description

Updates an norm_t** object, if it's not done so in a previous phase.

Todo

If the norm_t** object is already up to data, this step is a waste of time. Consider using an altnerative approach.

Author

Ole Kristian Ekseth (oekseth)

Date

21.12.2010 by oekseth (initial)
16.09.2011 by oekseth (asserts)

The documentation for this class was generated from the following file:

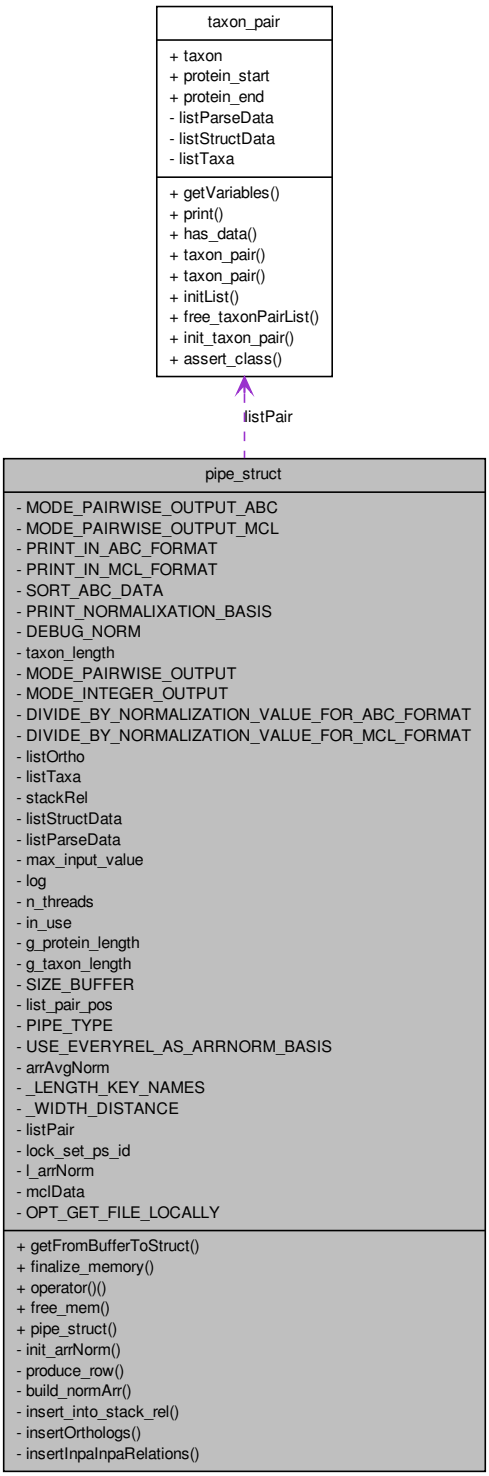
- pipe_norm.h

6.9 pipe_struct Class Reference

Either builds co-orthologs or builds the strings for the result file.

```
#include <pipe_struct.h>
```

Collaboration diagram for pipe_struct:



Public Member Functions

- void getFromBufferToStruct (const uint taxon_length, uint max_buffer_size, const bool only_inparalogs)

  Opens a buffer and dumps it to memory if the space for it is allocated.
- void finalize_memory (const uint taxon_length)

  Clears the memory allocated for this thread.
- void * operator() (void *item)

  The method of parallisation.
- void free_mem ()

  De-allocates the memory reserved for this object.
- pipe_struct (uint _nthread, uint _size_prot, uint taxon_length, pipe_t type, bool _USE_EVERYREL_AS_ARRNORM_BASIS, norm_t **normArr, log_builder_t *_log, bool _MODE_PAIRWISE_OUTPUT, bool _MODE_INTEGER_OUTPUT, bool _DIVIDE_BY_NORMALIZATION_VALUE_FOR_ABC_FORMAT, bool _DIVIDE_BY_NORMALIZATION_VALUE_FOR_MCL_FORMAT, id_simil_list &_listOrtho, taxa_t *_listTaxa, stack_rel *&_stackRel, list_file_struct_t *&_listStructData, list_file_parse_t *&_listParseData, float _max_input_value, bool _MODE_PAIRWISE_OUTPUT_ABC, bool _MODE_PAIRWISE_OUTPUT_MCL, bool _PRINT_IN_ABC_FORMAT, bool _PRINT_IN_MCL_FORMAT, bool _SORT_ABC_DATA, bool _PRINT_NORMALIXATION_BASIS, bool _DEBUG_NORM)

  Constructor for the class.

6.9.1 Detailed Description

Either builds co-orthologs or builds the strings for the result file.

Returns

A 'bucket_norm object.

Author

Ole Kristian Ekseth (oekseth)

Date

21.12.2010 by oekseth (initial)
16.09.2011 by oekseth (asserts)
24.12.2011 by oekseth (removed calls to 'extern' variables to ease the inclusion of thisclass as a libary)
25.12.2011 by oekseth (cleanup).

6.9.2 Constructor & Destructor Documentation

6.9.2.1 pipe_struct::pipe_struct (uint *_nthread,* uint *_size_prot,* uint *taxon_length,* pipe_t *type,* bool *_USE_EVERYREL_AS_ARRNORM_BASIS,* norm_t ∗∗ *normArr,* log_builder_t ∗ *_log,* bool *_MODE_PAIRWISE_OUTPUT,* bool *_MODE_INTEGER_OUTPUT,* bool *_DIVIDE_BY_NORMALIZATION_VALUE_FOR_ABC_FORMAT,* bool *_DIVIDE_BY_NORMALIZATION_VALUE_FOR_MCL_FORMAT,* id_simil_list & *_listOrtho,* taxa_t ∗ *_listTaxa,* stack_rel ∗& *_stackRel,* list_file_struct_t ∗& *_listStructData,* list_file_parse_t ∗& *_listParseData,* float *_max_input_value,* bool *_MODE_PAIRWISE_OUTPUT_ABC,* bool *_MODE_PAIRWISE_OUTPUT_MCL,* bool *_PRINT_IN_ABC_FORMAT,* bool *_PRINT_IN_MCL_FORMAT,* bool *_SORT_ABC_DATA,* bool *_PRINT_NORMALIXATION_BASIS,* bool *_DEBUG_NORM*)

Constructor for the class.

Remarks

Operation decided when setting the 'PIPE_TYPE' variabel (PIPE_TYPE == DUMP): Writes the data to a matrix, for further processing by another program (PIPE_-TYPE == INPA_ORTH) Creates inparalogs based upon orthologs. (PIPE_TYPE == INPA_INPA) Creates inparalogs based upon orthologs' inparalogs.

The documentation for this class was generated from the following file:

- pipe_struct.h

6.10 pipe_write Class Reference

Builds (writes) the result files, consisting of the strings given.

`#include <pipe_write.h>`

Public Member Functions

- void free_mem (bool SORT_ABC_DATA, char ∗FILE_BINARY_LOCATION)
- void ∗ operator() (void ∗item)

  The method of parallisation.
- pipe_write (log_builder_t ∗_log, taxa_t ∗listTaxa, int taxon_length, char ∗FILE_-BINARY_LOCATION, bool PRINT_IN_ABC_FORMAT, bool PRINT_IN_MCL_FORMAT, mcl_t TYPE_OF_RESULTFILE_TO_STDOUT)

  The constructor:

6.10.1 Detailed Description

Builds (writes) the result files, consisting of the strings given.

Remarks

Processes in serial the block of strings sent from class 'pipe_struct'.

Author

Ole Kristian Ekseth (oekseth)

Date

30.09.2009 by oekseth (initial)
16.09.2011 by oekseth (asserts)

6.10.2 Member Function Documentation

6.10.2.1 void pipe_write::free_mem (bool *SORT_ABC_DATA*, char * *FILE_BINARY_LOCATION*)

! Closes the files, adding the final trailings to the mcl files. De-allocates memory reserved.

The documentation for this class was generated from the following file:

- pipe_write.h

6.11 taxon_pair Class Reference

Defines the next protein to work on.

`#include <taxon_pair.h>`

Public Member Functions

- void getVariables (uint &_taxon, uint &_protein_start, uint &_protein_end)

  Returns the variables:
- void print ()

  Prints the data.
- bool has_data ()

  Returns true if is has more data.
- taxon_pair (uint _taxon, uint _protein_start, uint _protein_end, list_file_parse_t *&_listParseData, list_file_struct_t *&_listStructData, taxa_t *_listTaxa)

  The constructor.
- taxon_pair ()

  The constructor.

Static Public Member Functions

- static taxon_pair * initList (uint size)

  Returns an allocated region: A standard interface to ensure consistency.
- static void free_taxonPairList (taxon_pair *list)

A standard interface to ensure consistency:

- static taxon_pair ∗ init_taxon_pair (uint taxon_start, uint taxon_end, uint taxon_-length, const bool only_inpa, const bool from_parse, list_file_parse_t ∗listParseData, list_file_struct_t ∗&listStructData, taxa_t ∗listTaxa)

  Builds the set of blocks to be used during the parsing.
- static void assert_class (const bool print_info)

  The main test function for this class.

Public Attributes

- uint taxon

  The taxon id of interest.
- uint protein_start

  The first protein of this collection.
- uint protein_end

  The last protein of this collection.

6.11.1 Detailed Description

Defines the next protein to work on.

Author

Ole Kristian Ekseth (oekseth)

Date

18.03.2011 by oekseth (initial)
16.09.2011 by oekseth (asserts)
24.12.2011 by oekseth (removed calls to 'extern' variables to ease the inclusion of this class as a libary)
31.12.2011 (cleanup)

The documentation for this class was generated from the following file:

- taxon_pair.h

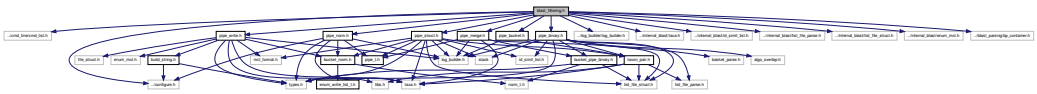
Chapter 7

File Documentation

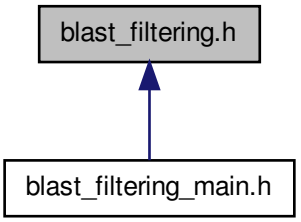
7.1 blast_filtering.h File Reference

```
#include "../cmd_line/cmd_list.h"
#include "../configure.h"
#include "../log_builder/log_builder.h"
#include "../internal_blast/taxa.h"
#include "../internal_blast/id_simil_list.h"
#include "../internal_blast/list_file_parse.h"
#include "../internal_blast/list_file_struct.h"
#include "../internal_blast/enum_mcl.h"
#include "../blast_parsing/bp_container.h"
#include "pipe_bucket.h"
#include "pipe_binary.h"
#include "pipe_merge.h"
#include "pipe_norm.h"
#include "pipe_write.h"
#include "pipe_struct.h"
```

Include dependency graph for blast_filtering.h:



This graph shows which files directly or indirectly include this file:



Classes

- class blast_filtering

  Produces a filtered output of the input.

Typedefs

- typedef class blast_filtering blast_filtering_t

  Produces a filtered output of the input.

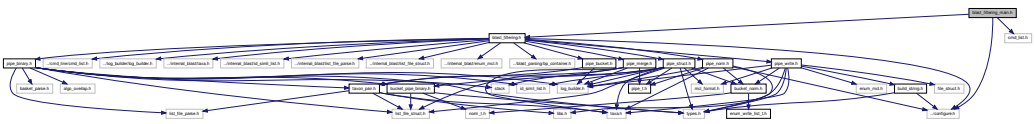
7.1.1 Detailed Description

7.2 blast_filtering_main.h File Reference

The launcher of the filtering library.

```
#include "../configure.h"
#include "blast_filtering.h"
#include "cmd_list.h"
```

Include dependency graph for blast_filtering_main.h:



7.2.1 Detailed Description

The launcher of the filtering library.

Author

Ole Kristian Ekseth (oekseth)

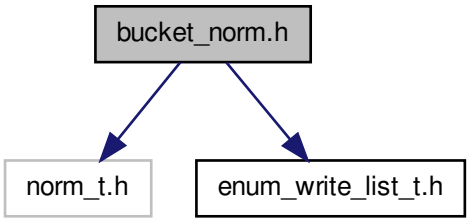
Date

31.12.2011 by oekseth (initial)

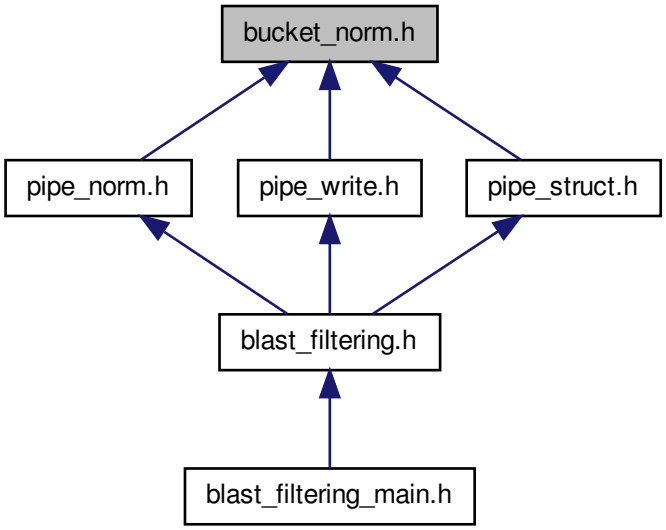
7.3 bucket_norm.h File Reference

```
#include "norm_t.h"
#include "enum_write_list_t.h"
```

Include dependency graph for bucket_norm.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct bucket_norm

  Data container holding changes in the normative array.

Typedefs

- typedef struct bucket_norm bucket_norm_t

  Data container holding changes in the normative array.

7.3.1 Detailed Description

7.3.2 Typedef Documentation

7.3.2.1 typedef struct bucket_norm bucket_norm_t

Data container holding changes in the normative array.

Author

Ole Kristian Ekseth (oekseth)

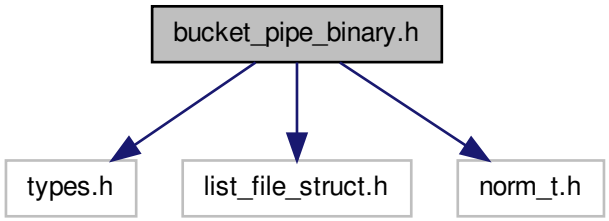
Date

02.11.2011 by oekseth (initial)

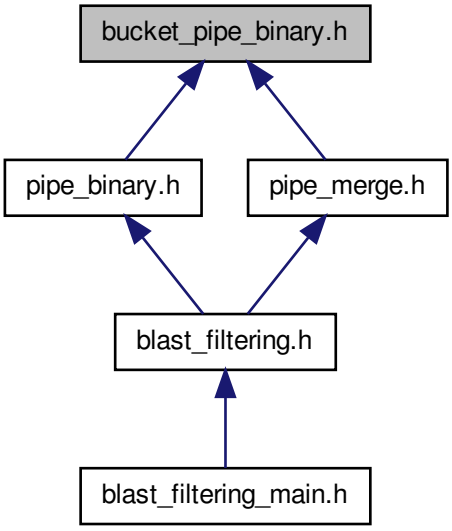
7.4 bucket_pipe_binary.h File Reference

```
#include "types.h"
#include "list_file_struct.h"
#include "norm_t.h"
```

Include dependency graph for bucket_pipe_binary.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct bucket_pipe_binary

  A transport container.

Typedefs

- typedef struct bucket_pipe_binary bucket_pipe_binary_t

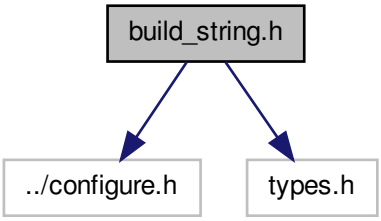
  A transport container.

7.4.1 Detailed Description

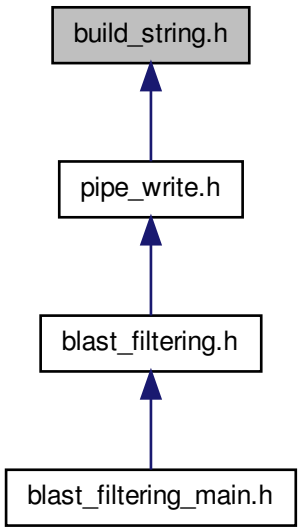
7.5 build_string.h File Reference

```
#include "../configure.h"
#include "types.h"
```

Include dependency graph for build_string.h:



This graph shows which files directly or indirectly include this file:



Classes

- class build_string

  Produces a row of chars for the protein given.

Typedefs

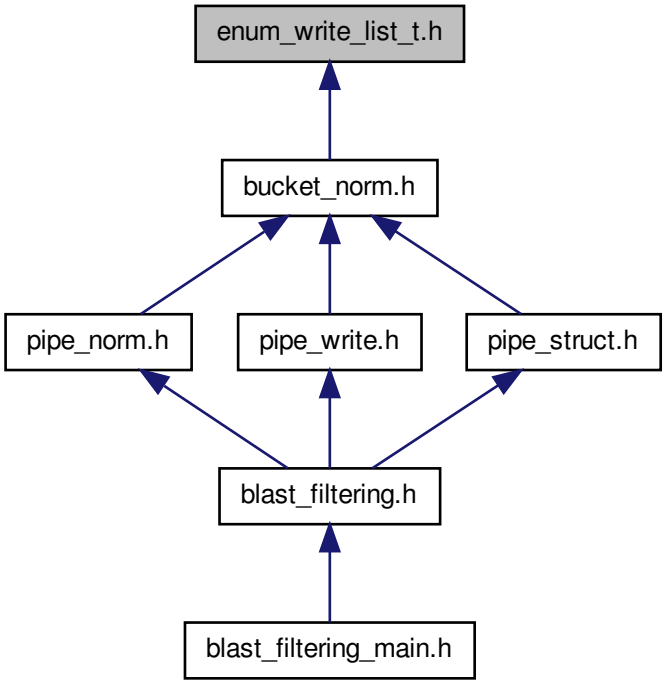
- typedef build_string build_string_t

  Produces a row of chars for the protein given.

7.5.1 Detailed Description

7.6 enum_write_list_t.h File Reference

This graph shows which files directly or indirectly include this file:



Typedefs

- typedef enum write_list write_list_t

  In order to identify the files.

Enumerations

- enum write_list { **blast_ortho_pairs_list_numbers**, **blast_inparalog_list_numbers**, **blast_complete_list_numbers**, **blast_ortho_relations_list_numbers** }

  In order to identify the files.

Variables

- static const uint size_write_list_t = 4

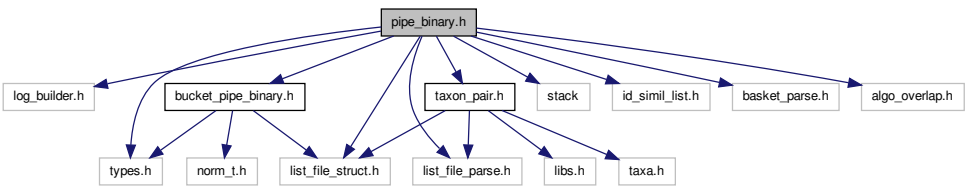
  The number of elements in the enum write_list.

7.6.1 Detailed Description

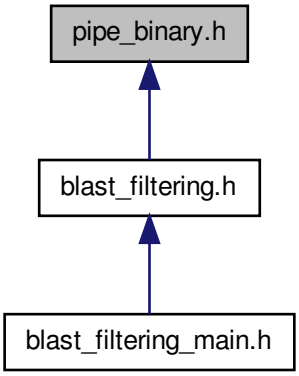
7.7 pipe_binary.h File Reference

```
#include "log_builder.h"
#include "types.h"
#include "list_file_parse.h"
#include "list_file_struct.h"
#include <stack>
#include "bucket_pipe_binary.h"
#include "taxon_pair.h"
#include "id_simil_list.h"
#include "basket_parse.h"
#include "algo_overlap.h"
```

Include dependency graph for pipe_binary.h:



This graph shows which files directly or indirectly include this file:



Classes

- class pipe_binary

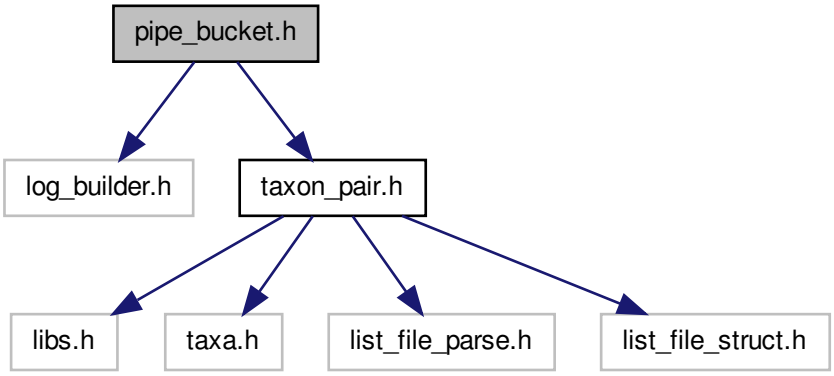
  Filters orthologs- and inparalogs in parallel.

7.7.1 Detailed Description

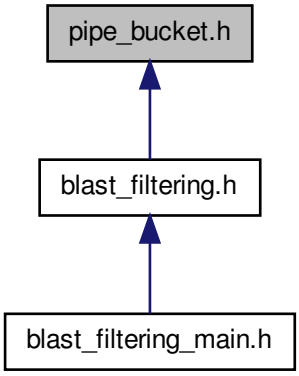
7.8 pipe_bucket.h File Reference

```
#include "log_builder.h"
#include "taxon_pair.h"
```

Include dependency graph for pipe_bucket.h:



This graph shows which files directly or indirectly include this file:



Classes

- class pipe_bucket

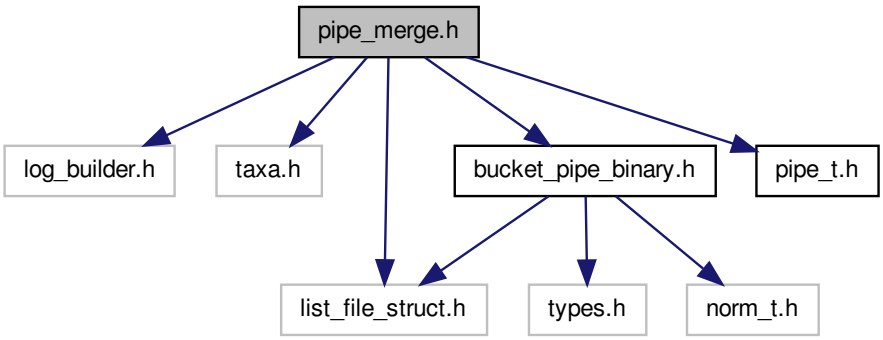
  Produces buckets of numbered items to parse.

7.8.1 Detailed Description

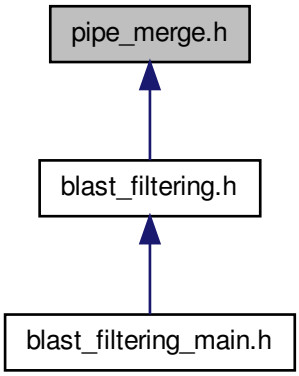
7.9 pipe_merge.h File Reference

```
#include "log_builder.h"
#include "taxa.h"
#include "list_file_struct.h"
#include "bucket_pipe_binary.h"
#include "pipe_t.h"
```

Include dependency graph for pipe_merge.h:



This graph shows which files directly or indirectly include this file:



Classes

- class pipe_merge

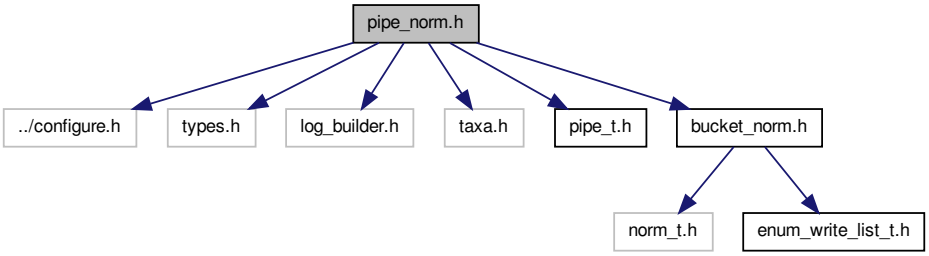
  Merges containers building a filtered set of orthologs- and inparalogs.

7.9.1 Detailed Description

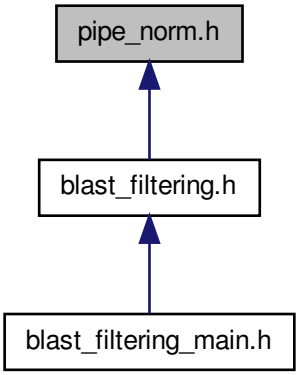
7.10 pipe_norm.h File Reference

```
#include "../configure.h"
#include "types.h"
#include "log_builder.h"
#include "taxa.h"
#include "pipe_t.h"
#include "bucket_norm.h"
```

Include dependency graph for pipe_norm.h:



This graph shows which files directly or indirectly include this file:



Classes

- class pipe_norm

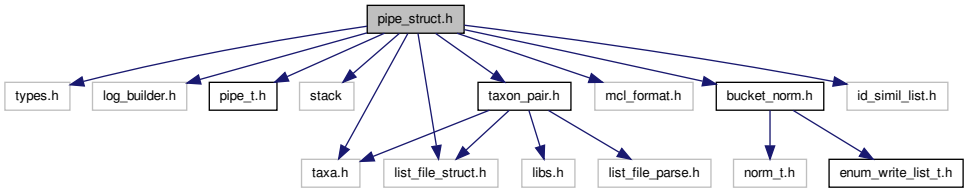
  *Updates an norm_t** object, if it's not done so in a previous phase.*

7.10.1 Detailed Description

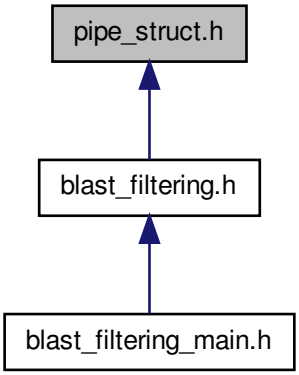
7.11 pipe_struct.h File Reference

```
#include "types.h"
#include "log_builder.h"
#include "pipe_t.h"
#include <stack>
#include "taxa.h"
#include "list_file_struct.h"
#include "mcl_format.h"
#include "taxon_pair.h"
#include "bucket_norm.h"
#include "id_simil_list.h"
```

Include dependency graph for pipe_struct.h:



This graph shows which files directly or indirectly include this file:



Classes

- class pipe_struct

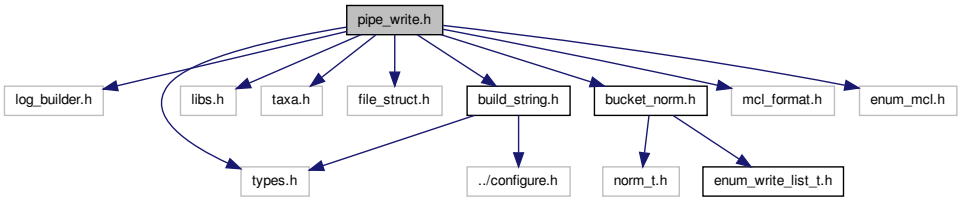
  Either builds co-orthologs or builds the strings for the result file.

7.11.1 Detailed Description

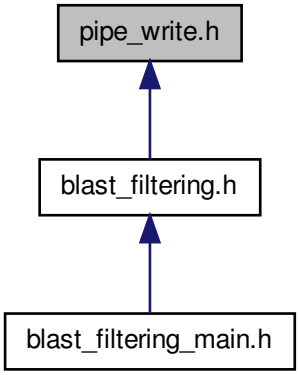
7.12 pipe_write.h File Reference

```
#include "log_builder.h"
#include "types.h"
#include "libs.h"
#include "taxa.h"
#include "file_struct.h"
#include "build_string.h"
#include "bucket_norm.h"
#include "mcl_format.h"
#include "enum_mcl.h"
```

Include dependency graph for pipe_write.h:



This graph shows which files directly or indirectly include this file:



Classes

- class pipe_write

  Builds (writes) the result files, consisting of the strings given.

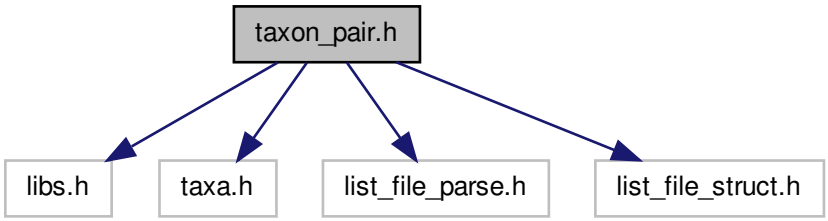
7.12.1 Detailed Description

7.13 taxon_pair.h File Reference

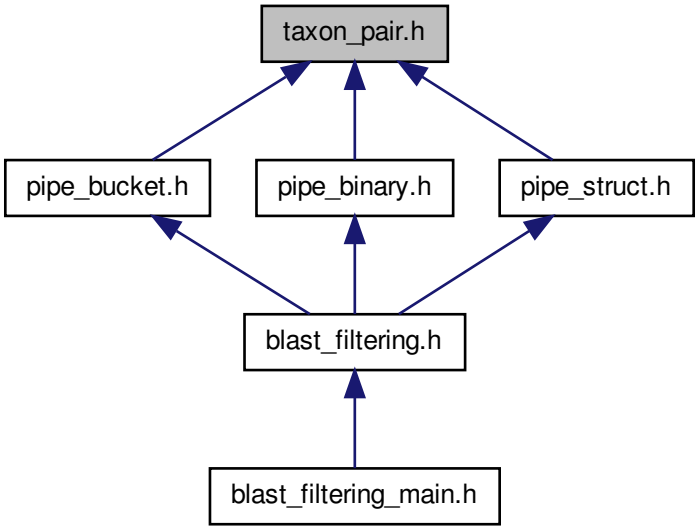
```
#include "libs.h"
```

```
#include "taxa.h"
#include "list_file_parse.h"
#include "list_file_struct.h"
```

Include dependency graph for taxon_pair.h:



This graph shows which files directly or indirectly include this file:



Classes

- class taxon_pair

  Defines the next protein to work on.

Typedefs

- typedef class taxon_pair taxon_pair_t

  Defines the next protein to work on.

7.13.1 Detailed Description

Index