

EAGLER

Generated by Doxygen 1.8.13

Contents

1	Deprecated List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	bases Namespace Reference	11
6.1.1	Detailed Description	11
6.1.2	Function Documentation	11
6.1.2.1	count_bases() [1/2]	11
6.1.2.2	count_bases() [2/2]	12
6.2	read_type Namespace Reference	12
6.2.1	Enumeration Type Documentation	12
6.2.1.1	ReadType	13
6.2.2	Function Documentation	13
6.2.2.1	string_to_read_type()	13
6.3	scaffolder Namespace Reference	13

6.3.1	Detailed Description	14
6.3.2	Function Documentation	14
6.3.2.1	extend_contig()	14
6.3.2.2	extend_contig_poa()	15
6.3.2.3	find_possible_extensions() [1/2]	15
6.3.2.4	find_possible_extensions() [2/2]	16
6.3.2.5	get_extension_mv_realign()	16
6.3.2.6	get_extension_mv_simple() [1/2]	17
6.3.2.7	get_extension_mv_simple() [2/2]	17
6.3.2.8	set_inner_margin()	17
6.3.2.9	set_max_extension_len()	17
6.3.2.10	set_min_coverage()	18
6.3.2.11	set_outer_margin()	18
6.3.3	Variable Documentation	18
6.3.3.1	inner_margin	18
6.3.3.2	max_ext_length	19
6.3.3.3	min_coverage	19
6.3.3.4	outer_margin	19
6.3.3.5	tmp_contig_file	19
6.3.3.6	tmp_reads_file	19
6.3.3.7	tmp_sam_file	19
6.4	utility Namespace Reference	20
6.4.1	Function Documentation	21
6.4.1.1	base_to_idx()	21
6.4.1.2	CharString_to_string()	21
6.4.1.3	contributes_to_contig_len()	22
6.4.1.4	contributes_to_seq_len()	22
6.4.1.5	create_seq_id()	22
6.4.1.6	Dna5String_to_string()	23
6.4.1.7	execute_command()	23

6.4.1.8	exit_with_message()	24
6.4.1.9	get_concurrency_level()	24
6.4.1.10	idx_to_base()	24
6.4.1.11	is_command_available()	25
6.4.1.12	map_alignments()	25
6.4.1.13	read_fasta()	26
6.4.1.14	read_sam() [1/2]	26
6.4.1.15	read_sam() [2/2]	26
6.4.1.16	reverse_complement()	27
6.4.1.17	set_concurrency_level()	27
6.4.1.18	throw_exception()	27
6.4.1.19	write_fasta() [1/2]	28
6.4.1.20	write_fasta() [2/2]	28
6.4.2	Variable Documentation	28
6.4.2.1	command_buffer	29
6.4.2.2	error_buffer	29
6.4.2.3	hardware_concurrency	29
6.4.2.4	seq_id_buffer	29
7	Class Documentation	31
7.1	Aligner Class Reference	31
7.1.1	Detailed Description	32
7.1.2	Constructor & Destructor Documentation	32
7.1.2.1	Aligner()	32
7.1.2.2	~Aligner()	33
7.1.3	Member Function Documentation	33
7.1.3.1	align() [1/4]	33
7.1.3.2	align() [2/4]	33
7.1.3.3	align() [3/4]	34
7.1.3.4	align() [4/4]	34
7.1.3.5	get_instance()	34

7.1.3.6	get_name()	34
7.1.3.7	get_tmp_alignment_filename()	34
7.1.3.8	get_tmp_contig_filename()	34
7.1.3.9	get_tmp_reference_filename()	34
7.1.3.10	index()	34
7.1.3.11	init()	35
7.1.4	Member Data Documentation	35
7.1.4.1	name	35
7.1.4.2	tech_type	35
7.2	bases::BasesCounter Class Reference	35
7.2.1	Detailed Description	36
7.2.2	Constructor & Destructor Documentation	36
7.2.2.1	BasesCounter()	36
7.2.3	Member Function Documentation	36
7.2.3.1	digest_base()	36
7.2.3.2	refresh_stats()	37
7.2.4	Member Data Documentation	37
7.2.4.1	count	37
7.2.4.2	coverage	37
7.2.4.3	max_idx	37
7.3	BwaAligner Class Reference	38
7.3.1	Detailed Description	39
7.3.2	Constructor & Destructor Documentation	39
7.3.2.1	BwaAligner()	39
7.3.3	Member Function Documentation	39
7.3.3.1	align() [1/4]	39
7.3.3.2	align() [2/4]	39
7.3.3.3	align() [3/4]	40
7.3.3.4	align() [4/4]	40
7.3.3.5	index()	41

7.4	Connector Class Reference	41
7.4.1	Detailed Description	42
7.4.2	Constructor & Destructor Documentation	42
7.4.2.1	Connector()	42
7.4.2.2	~Connector()	42
7.4.3	Member Function Documentation	42
7.4.3.1	connect_contigs()	42
7.4.3.2	dump_scaffolds()	43
7.4.3.3	get_scaffolds()	43
7.5	Contig Class Reference	43
7.5.1	Detailed Description	44
7.5.2	Constructor & Destructor Documentation	45
7.5.2.1	Contig() [1/3]	45
7.5.2.2	Contig() [2/3]	45
7.5.2.3	Contig() [3/3]	45
7.5.3	Member Function Documentation	46
7.5.3.1	dump_anchors()	46
7.5.3.2	ext_left()	46
7.5.3.3	ext_right()	46
7.5.3.4	id()	47
7.5.3.5	left_id()	47
7.5.3.6	operator!=(())	47
7.5.3.7	operator==(())	47
7.5.3.8	reverse_complement()	48
7.5.3.9	right_ext_pos()	48
7.5.3.10	right_id()	48
7.5.3.11	seq()	48
7.5.3.12	set_id()	48
7.5.3.13	total_ext_left()	49
7.5.3.14	total_ext_right()	49

7.5.3.15	<code>total_len()</code>	49
7.6	Extension Class Reference	50
7.6.1	Detailed Description	50
7.6.2	Constructor & Destructor Documentation	50
7.6.2.1	<code>Extension()</code>	50
7.6.3	Member Function Documentation	51
7.6.3.1	<code>curr_pos()</code>	51
7.6.3.2	<code>do_operation()</code>	51
7.6.3.3	<code>read_id()</code>	51
7.6.3.4	<code>seq()</code>	52
7.6.4	Member Data Documentation	52
7.6.4.1	<code>is_dropped</code>	52
7.7	GraphMapAligner Class Reference	52
7.7.1	Constructor & Destructor Documentation	53
7.7.1.1	<code>GraphMapAligner()</code>	53
7.7.1.2	<code>~GraphMapAligner()</code>	53
7.7.2	Member Function Documentation	54
7.7.2.1	<code>align()</code> [1/4]	54
7.7.2.2	<code>align()</code> [2/4]	54
7.7.2.3	<code>align()</code> [3/4]	54
7.7.2.4	<code>align()</code> [4/4]	54
7.7.2.5	<code>index()</code>	55
7.8	Scaffold Class Reference	55
7.8.1	Detailed Description	56
7.8.2	Constructor & Destructor Documentation	56
7.8.2.1	<code>Scaffold()</code>	56
7.8.3	Member Function Documentation	56
7.8.3.1	<code>add_contig()</code>	56
7.8.3.2	<code>contains()</code>	57
7.8.3.3	<code>first_contig()</code>	57
7.8.3.4	<code>get_combined_sequence()</code>	57
7.8.3.5	<code>get_contigs()</code>	58
7.8.3.6	<code>last_contig()</code>	58
7.8.3.7	<code>merge()</code>	58
7.8.3.8	<code>num_contigs()</code>	58
7.8.3.9	<code>trim()</code>	59

8	File Documentation	61
8.1	src/aligners/aligner.cpp File Reference	61
8.1.1	Detailed Description	61
8.2	src/aligners/aligner.h File Reference	62
8.2.1	Detailed Description	63
8.3	src/aligners/bwa.cpp File Reference	63
8.3.1	Detailed Description	63
8.4	src/aligners/bwa.h File Reference	64
8.4.1	Detailed Description	64
8.5	src/aligners/graphmap.cpp File Reference	65
8.5.1	Detailed Description	65
8.6	src/aligners/graphmap.h File Reference	65
8.6.1	Detailed Description	66
8.7	src/bases.cpp File Reference	67
8.7.1	Detailed Description	67
8.8	src/bases.h File Reference	68
8.8.1	Detailed Description	69
8.8.2	Macro Definition Documentation	69
8.8.2.1	NUM_BASES	69
8.8.3	Typedef Documentation	69
8.8.3.1	bool_predicate	70
8.9	src/connector.cpp File Reference	70
8.9.1	Detailed Description	70
8.10	src/connector.h File Reference	71
8.10.1	Detailed Description	72
8.10.2	Macro Definition Documentation	72
8.10.2.1	ANCHOR_THRESHOLD	72
8.10.2.2	COMPLEMENT	72
8.10.2.3	SECONDARY_LINE	72
8.10.2.4	UNMAPPED	73

8.11	src/contig.cpp File Reference	73
8.11.1	Detailed Description	73
8.12	src/contig.h File Reference	74
8.12.1	Detailed Description	75
8.12.2	Macro Definition Documentation	75
8.12.2.1	ANCHOR_LEN	75
8.13	src/extension.cpp File Reference	75
8.13.1	Detailed Description	76
8.14	src/extension.h File Reference	76
8.14.1	Detailed Description	77
8.14.2	Enumeration Type Documentation	77
8.14.2.1	Operation	77
8.15	src/main.cpp File Reference	77
8.15.1	Detailed Description	79
8.15.2	Macro Definition Documentation	79
8.15.2.1	PATH_BUFFER_SIZE	79
8.15.2.2	RELEASE_DATE	79
8.15.2.3	VERSION	79
8.15.3	Function Documentation	79
8.15.3.1	main()	79
8.15.3.2	set_output_paths()	79
8.15.3.3	setup_cmd_interface()	80
8.15.4	Variable Documentation	80
8.15.4.1	contigs_filename	80
8.15.4.2	draft_genome_filename	80
8.15.4.3	extensions_filename	80
8.15.4.4	reads_filename	80
8.15.4.5	scaffolds_filename	80
8.15.4.6	tmp_dirname	80
8.15.4.7	trim_circular_genome	81

8.15.4.8	use_graphmap_aligner	81
8.15.4.9	use_POA_consensus	81
8.15.4.10	use_tech_type	81
8.16	src/scaffold.cpp File Reference	81
8.16.1	Detailed Description	82
8.17	src/scaffold.h File Reference	82
8.17.1	Detailed Description	83
8.18	src/scaffolder.cpp File Reference	83
8.18.1	Detailed Description	85
8.18.2	Macro Definition Documentation	85
8.18.2.1	INNER_MARGIN	85
8.18.2.2	MIN_COVERAGE	85
8.18.2.3	OUTER_MARGIN	85
8.19	src/scaffolder.h File Reference	86
8.19.1	Detailed Description	87
8.20	src/utility.cpp File Reference	87
8.20.1	Detailed Description	89
8.21	src/utility.h File Reference	89
8.21.1	Detailed Description	91
8.21.2	Macro Definition Documentation	92
8.21.2.1	COMMAND_BUFFER_SIZE	92
8.21.2.2	DEBUG	92
8.21.2.3	DEBUG_BLOCK	92
8.21.2.4	DEBUG_VAR	92
8.21.2.5	ERROR_BUFFER_SIZE	92
8.21.2.6	MINIMUM_CONTIG_LEN	92
8.21.2.7	SEQ_ID_BUFFER_SIZE	93
8.21.2.8	UNMAPPED	93
8.21.3	Typedef Documentation	93
8.21.3.1	AlignmentCollection	93

Chapter 1

Deprecated List

Member [BwaAligner::align](#) (const CharString &id, const Dna5String &contig, const char *reads_filename)

Member [scaffolder::get_extension_mv_simple](#) (const vector< string > &extensions)

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

bases	Namespace for BasesCounter class and supporting functions	11
read_type	12
scaffolder	Scaffolder namespace provides functions for different methods of contig extension	13
utility	20

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Aligner	31
BwaAligner	38
GraphMapAligner	52
bases::BasesCounter	35
Connector	41
Contig	43
Extension	50
Scaffold	55

Chapter 4

Class Index

4.1 Class List

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

Aligner	Class representing an abstract aligner	31
bases::BasesCounter	Used for calculating various statistics at specific positions in the contig extension process . . .	35
BwaAligner	38
Connector	Connector class	41
Contig	Contig class represent contig sequences	43
Extension	Extension class	50
GraphMapAligner	52
Scaffold	Scaffold class is representation of scaffold structure	55

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

src/ bases.cpp	Implementation file for the BasesCounter class and associated factory functions	67
src/ bases.h	Header file for BasesCounter class and various functions which calculate statistics over a sequence of bases	68
src/ connector.cpp	Implementation file for Connector class	70
src/ connector.h	Header file for Connector class	71
src/ contig.cpp	Implementation file for Contig class	73
src/ contig.h	Header file for Contig class	74
src/ extension.cpp	Implementation file for Extension class	75
src/ extension.h	Header file for Extension class	76
src/ main.cpp	Entry point of program and main program of project	77
src/ scaffold.cpp	Implementation file for Scaffold class	81
src/ scaffold.h	Header file for Scaffold class	82
src/ scaffolder.cpp	Implementation file for scaffolder namespace	83
src/ scaffolder.h	Header file for scaffolder namespace	86
src/ utility.cpp	Implementation of various utility functions	87
src/ utility.h	Various utility functions	89
src/aligners/ aligner.cpp	Implementation file for the abstract Aligner class	61
src/aligners/ aligner.h	Declaration of an abstract aligner class	62

src/aligners/ bwa.cpp	
Implementation file for aligner namespace	63
src/aligners/ bwa.h	
Header file for aligner namespace	64
src/aligners/ graphmap.cpp	65
src/aligners/ graphmap.h	
Declaration of the GraphMap class	65

Chapter 6

Namespace Documentation

6.1 bases Namespace Reference

Namespace for [BasesCounter](#) class and supporting functions.

Classes

- class [BasesCounter](#)

Used for calculating various statistics at specific positions in the contig extension process.

Functions

- [BasesCounter count_bases](#) (const vector< shared_ptr< [Extension](#) >> &extensions, [bool_predicate](#) is_
read_eligible, int offset)
Count bases at a specific position in the given extensions.
- [BasesCounter count_bases](#) (const vector< shared_ptr< [Extension](#) >> &extensions)
Count bases at specific position in extensions.

6.1.1 Detailed Description

Namespace for [BasesCounter](#) class and supporting functions.

6.1.2 Function Documentation

6.1.2.1 count_bases() [1/2]

```
BasesCounter bases::count_bases (
    const vector< shared_ptr< Extension >> & extensions,
    bool\_predicate is_read_eligible,
    int offset )
```

Count bases at a specific position in the given extensions.

Creates a [BasesCounter](#) object and digests one base from each extending sequence.

Parameters

<i>extensions</i>	Extension sequences, each extension stores the index of the first unprocessed index in it's sequence
<i>is_read_eligible</i>	lambda function called over the active base of each extension, should return true if the base should be processed
<i>offset</i>	the offset from current index in all Extension objects of the base to be digested

Returns

[BasesCounter](#) object with summarized data from one base from each extension

6.1.2.2 `count_bases()` [2/2]

```
BasesCounter bases::count_bases (
    const vector< shared_ptr< Extension >> & extensions )
```

Count bases at specific position in extensions.

Wrapper method for other count_bases method. All extensions are eligible for usage.

Parameters

<i>extensions</i>	Extension sequences, each extension stores the index of the first unprocessed index in it's sequence
-------------------	--

Returns

[BasesCounter](#) object with summarized data from one base from each extension

6.2 `read_type` Namespace Reference

Enumerations

- enum [ReadType](#) { [PacBio](#), [ONT](#) }
Enum used to distinguish read types.

Functions

- [ReadType string_to_read_type](#) (const char *read_type)
Converts the given string to a ReadType enumerator object.

6.2.1 Enumeration Type Documentation

6.2.1.1 ReadType

```
enum read_type::ReadType
```

Enum used to distinguish read types.

Enumerator

PacBio	
ONT	

6.2.2 Function Documentation

6.2.2.1 string_to_read_type()

```
read_type::ReadType read_type::string_to_read_type (
    const char * read_type )
```

Converts the given string to a ReadType enumerator object.

Parameters

<code>read_type</code>	C-style string ID of the read type
------------------------	------------------------------------

Returns

the enum associated to the passed string

6.3 scaffolder Namespace Reference

Scaffolder namespace provides functions for different methods of contig extension.

Functions

- void `set_max_extension_len` (int length)
Methods sets maximum extension length.
- void `set_inner_margin` (int margin)
Sets the inner margin.
- void `set_outer_margin` (int margin)
Sets the outer margin.
- void `set_min_coverage` (int coverage)
Method sets the minimum coverage.

- void `find_possible_extensions` (const vector< BamAlignmentRecord > &aln_records, vector< shared_ptr< Extension >> *pleft_ext_reads, vector< shared_ptr< Extension >> *pright_ext_reads, const unordered_map< string, uint32_t > &read_name_to_id, uint64_t contig_len)
- string `get_extension_mv_simple` (const vector< shared_ptr< Extension >> &extensions)
- string `get_extension_mv_realign` (const vector< shared_ptr< Extension >> &extensions)
Method finds contig extension using majority vote on each position of possible extensions while coverage $\geq k$ and tries to realign reads which base isn't elected as majority.
- Contig * `extend_contig` (Dna5String &contig_seq, const vector< BamAlignmentRecord > &aln_records, const unordered_map< string, uint32_t > &read_name_to_id, const StringSet< CharString > &read_ids, const StringSet< Dna5String > &read_seqs)
Method tries to extend contig using global realignment method on both sides with given alignment records.
- Contig * `extend_contig_poa` (const Dna5String &contig_seq, const vector< BamAlignmentRecord > &aln_records, const unordered_map< string, uint32_t > &read_name_to_id)
Method tries to extend contig using POA consensus method on both sides with given alignment records.
- void `find_possible_extensions` (const vector< BamAlignmentRecord > &aln_records, vector< string > *pleft_extensions, vector< string > *pright_extensions, uint64_t contig_len)
Method finds substrings of reads which extend contig on both ends.
- string `get_extension_mv_simple` (const vector< string > &extensions)
Method finds contig extension using majority vote on each position of possible extensions while coverage $\geq k$.

Variables

- int `max_ext_length` = 1000
- int `inner_margin` = 5
- int `outer_margin` = 15
- int `min_coverage` = 5
- const char * `tmp_contig_file` = "tmp/extend_contig.fasta"
- const char * `tmp_reads_file` = "tmp/realign_reads.fasta"
- const char * `tmp_sam_file` = "tmp/realign.sam"

6.3.1 Detailed Description

Scaffolder namespace provides functions for different methods of contig extension.

Scaffolder namespace provides functions for different methods of contig extension. Simple majority vote method, local realignment with global realignment method and POA consensus method.

6.3.2 Function Documentation

6.3.2.1 extend_contig()

```
Contig * scaffolder::extend_contig (
    Dna5String & contig_seq,
    const vector< BamAlignmentRecord > & aln_records,
    const unordered_map< string, uint32_t > & read_name_to_id,
    const StringSet< CharString > & read_ids,
    const StringSet< Dna5String > & read_seqs )
```

Method tries to extend contig using global realignment method on both sides with given alignment records.

This process is iterative - in each step, after contig extension is found using local realignment method, dropped reads are globally realigned and new possible extensions of already extended contig are found. If the coverage of left and right possible extensions are both below the minimum coverage, contig cannot be extended anymore and process is stopped.

Parameters

<i>contig_seq</i>	the Sequence of the contig to be extended
<i>aln_records</i>	Alignment records from SAM file
<i>read_name_to_id</i>	Mapping from read name to integer ID.
<i>read_ids</i>	Reads names / string IDs.
<i>read_seqs</i>	Reads sequences.

Returns

[Contig](#) extended on both sides

6.3.2.2 extend_contig_poa()

```
Contig * scaffolder::extend_contig_poa (
    const Dna5String & contig_seq,
    const vector< BamAlignmentRecord > & aln_records,
    const unordered_map< string, uint32_t > & read_name_to_id )
```

Method tries to extend contig using POA consensus method on both sides with given alignment records.

Using POA generated consensus to extend contigs is actually a very simple idea. Local realignment method is used to get possible extension reads, substrings of defined length are created for each extension and these substrings are passed to the POA algorithm for generating consensus of these sequences. Afterwards, consensus sequences are appended to contig as left and right extensions.

Parameters

<i>contig_seq</i>	the Sequence of the contig to be extended
<i>aln_records</i>	Alignment records from SAM file
<i>read_name_to_id</i>	Mapping from read name to integer ID.

Returns

[Contig](#) extended on both sides.

6.3.2.3 find_possible_extensions() [1/2]

```
void scaffolder::find_possible_extensions (
    const vector< BamAlignmentRecord > & aln_records,
    vector< shared_ptr< Extension >> & pleft_ext_reads,
    vector< shared_ptr< Extension >> & pright_ext_reads,
    const unordered_map< string, uint32_t > & read_name_to_id,
    uint64_t contig_len )
```

6.3.2.4 find_possible_extensions() [2/2]

```
void scaffolder::find_possible_extensions (
    const vector< BamAlignmentRecord > & aln_records,
    vector< string > * pleft_extensions,
    vector< string > * pright_extensions,
    uint64_t contig_len )
```

Method finds substrings of reads which extend contig on both ends.

Every record/read mapped to contig needs to be checked if it is a possible extension. A record is suitable for extending contig if it is soft clipped and clipped part extends left of contig start, or if it is soft clipped and clipped part extends right of contig start. Additional criteria is introduced to check if alignment record, i.e. read, is feasible for contig extension and these are inner and outer margin. Reads whose alignment starts (left extension) or ends (right extension) within the inner margin is immediately suitable for extension. Reads whose alignment starts or ends within the outer margin are called dropped reads and are later used in global realignment method.

Parameters

<i>aln_records</i>	Records from SAM file
<i>pleft_extensions</i>	Pointer to possible left end extensions
<i>pright_extensions</i>	Pointer to possible right end extensions
<i>contig_len</i>	Length of contig

6.3.2.5 get_extension_mv_realign()

```
string scaffolder::get_extension_mv_realign (
    const vector< shared_ptr< Extension >> & extensions )
```

Method finds contig extension using majority vote on each position of possible extensions while coverage $\geq k$ and tries to realign reads which base isn't elected as majority.

Variation of simple extension majority vote method is to locally realign reads which bases at current extension position are not the same as base calculated with majority vote. Local realignment "looks" one move ahead, and checks if one of the alignment operations (match, mismatch, deletion, insertion) could correct read alignment to (extended) contig. "Looking ahead" is done by calculating the next base by majority vote, but only reads with correct base at current position are considered eligible for counting.

Parameters

<i>extensions</i>	Possible contig extensions.
-------------------	-----------------------------

Returns

Resulting contig extension.

6.3.2.6 `get_extension_mv_simple()` [1/2]

```
string scaffolder::get_extension_mv_simple (
    const vector< string > & extensions )
```

Method finds contig extension using majority vote on each position of possible extensions while coverage $\geq k$.

Deprecated

Parameters

<i>extensions</i>	Possible contig extensions strings
-------------------	------------------------------------

Returns

[Contig](#) extension.

6.3.2.7 `get_extension_mv_simple()` [2/2]

```
string scaffolder::get_extension_mv_simple (
    const vector< shared_ptr< Extension >> & extensions )
```

6.3.2.8 `set_inner_margin()`

```
void scaffolder::set_inner_margin (
    int margin )
```

Sets the inner margin.

Sets the value in BP for the margin of a read to be considered directly in the extension base computation.

Parameters

<i>margin</i>	the desired inner margin in base pairs
---------------	--

6.3.2.9 `set_max_extension_len()`

```
void scaffolder::set_max_extension_len (
    int length )
```

Methods sets maximum extension length.

Method limits length of extension to value given as parameter for all methods.

Parameters

<i>length</i>	Maximum extension length.
---------------	---------------------------

6.3.2.10 set_min_coverage()

```
void scaffolder::set_min_coverage (
    int coverage )
```

Method sets the minimum coverage.

Sets the minimum numbers of reads that have to be present in the majority vote to output an extension base.

Parameters

<i>coverage</i>	the desired coverage
-----------------	----------------------

6.3.2.11 set_outer_margin()

```
void scaffolder::set_outer_margin (
    int margin )
```

Sets the outer margin.

Sets the value in BP for the margin of a read to be considered for potential global realignment.

Parameters

<i>margin</i>	the desired outer margin in base pairs
---------------	--

6.3.3 Variable Documentation**6.3.3.1 inner_margin**

```
int scaffolder::inner_margin = 5
```

6.3.3.2 max_ext_length

```
int scaffolder::max_ext_length = 1000
```

6.3.3.3 min_coverage

```
int scaffolder::min_coverage = 5
```

6.3.3.4 outer_margin

```
int scaffolder::outer_margin = 15
```

6.3.3.5 tmp_contig_file

```
const char* scaffolder::tmp_contig_file = "tmp/extend_contig.fasta"
```

6.3.3.6 tmp_reads_file

```
const char* scaffolder::tmp_reads_file = "tmp/realign_reads.fasta"
```

6.3.3.7 tmp_sam_file

```
const char* scaffolder::tmp_sam_file = "tmp/realign.sam"
```

6.4 utility Namespace Reference

Functions

- unsigned int [get_concurrency_level](#) ()
Gets the concurrency level.
- void [set_concurrency_level](#) (int threads)
Sets the concurrency level.
- void [read_fasta](#) (StringSet< CharString > *pids, StringSet< Dna5String > *pseqs, char *ont_reads_↵ filename)
Reads a FASTA file.
- void [write_fasta](#) (const CharString &id, const Dna5String &seq, const char *filename)
Write a sequence to file.
- void [write_fasta](#) (const StringSet< CharString > &ids, const StringSet< Dna5String > &seqs, const char *filename)
Writes a set of sequences to file.
- void [read_sam](#) (BamHeader *pheader, vector< BamAlignmentRecord > *precords, const char *filename)
Read a SAM file.
- void [read_sam](#) (BamFileIn *pinput_file, BamHeader *pheader, vector< BamAlignmentRecord > *precords)
- void [map_alignments](#) (const char *filename, [AlignmentCollection](#) *collection, const unordered_map< string, uint32_t > &contig_name_to_id)
Reads and map aligned reads to contigs.
- void [execute_command](#) (const char *format,...)
Run shell command.
- int [base_to_idx](#) (char base)
Char base to int id.
- char [idx_to_base](#) (int idx)
Base id to char base.
- void [exit_with_message](#) (const char *format,...)
Prints error message and exits.
- string [CharString_to_string](#) (const CharString &str)
Converts seqan::CharString to std::string.
- string [Dna5String_to_string](#) (const Dna5String &str)
Converts seqan::Dna5String to std::string.
- int [contributes_to_seq_len](#) (char c)
Method used to determine read length from cigar string.
- int [contributes_to_contig_len](#) (char c)
Method used to determine length of contig part to which read is aligned.
- string [reverse_complement](#) (const Dna5String &seq)
Method creates reverse complement from string given as parameter.
- string [create_seq_id](#) (const char *format,...)
Helper method for creating sequence ID.
- bool [is_command_available](#) (const char *command)
Checks if the given command is available through the system shell.
- template<typename T >
void [throw_exception](#) (const char *format,...)
Throw an exception.

Variables

- unsigned int `hardware_concurrency`
- char `command_buffer` [`COMMAND_BUFFER_SIZE`] = { 0 }
Buffer to hold shell command strings.
- char `error_buffer` [`ERROR_BUFFER_SIZE`] = { 0 }
Buffer to hold a description string when an error occurs.
- char `seq_id_buffer` [`SEQ_ID_BUFFER_SIZE`] = { 0 }
Buffer used to build the name of a sequence.

6.4.1 Function Documentation

6.4.1.1 `base_to_idx()`

```
int utility::base_to_idx (
    char base )
```

Char base to int id.

Converts a nucleotide character to its corresponding integer id. the mapping used is {'A': 0, 'T': 1, 'G': 2, 'C': 3}.

Parameters

<i>base</i>	nucleotide base
-------------	-----------------

Returns

integer base id

Exceptions

<i>std::invalid_argument</i>	when the base is not a valid nucleotide
------------------------------	---

6.4.1.2 `CharString_to_string()`

```
string utility::CharString_to_string (
    const CharString & str )
```

Converts `seqan::CharString` to `std::string`.

Parameters

<i>str</i>	String for type conversion.
------------	-----------------------------

Returns

Converted string.

6.4.1.3 contributes_to_contig_len()

```
int utility::contributes_to_contig_len (  
    char c )
```

Method used to determine length of contig part to which read is aligned.

Alignment operations which contribute to contig length are: alignment match, deletion from reference, soft clipping, sequence mismatch and sequence match.

Parameters

<i>c</i>	Character representing alignment operation.
----------	---

Returns

1 if contributes, 0 otherwise.

6.4.1.4 contributes_to_seq_len()

```
int utility::contributes_to_seq_len (  
    char c )
```

Method used to determine read length from cigar string.

Alignment operations which contribute to read length are: alignment match, insertion to reference, soft clipping, sequence mismatch and sequence match.

Parameters

<i>c</i>	Character representing alignment operation.
----------	---

Returns

1 if contributes, 0 otherwise.

6.4.1.5 create_seq_id()

```
string utility::create_seq_id (  
    const char * format,  
    ... )
```

Helper method for creating sequence ID.

Parameters

<i>format</i>	Printf style format string.
...	Printf style arguments to fill the format string.

Returns

Sequence ID.

6.4.1.6 Dna5String_to_string()

```
string utility::Dna5String_to_string (
    const Dna5String & str )
```

Converts seqan::Dna5String to std::string.

Parameters

<i>str</i>	String for type conversion
------------	----------------------------

Returns

Converted string

6.4.1.7 execute_command()

```
void utility::execute_command (
    const char * format,
    ... )
```

Run shell command.

Executes a shell command given as a combination of a format string and variable number of arguments to be inserted in the format string.

Parameters

<i>format</i>	the format of the command
...	printf style arguments to fill the format string

Exceptions

Exceptions

<code>std::runtime_error</code>	when the exit value of the command is not 0
---------------------------------	---

6.4.1.8 `exit_with_message()`

```
void utility::exit_with_message (
    const char * format,
    ... )
```

Prints error message and exits.

Prints the error tag [ERROR] and the given error message to stderr, than exits the program with exit code 1.

Parameters

<i>format</i>	the the format of the error message
...	printf style arguments to fill the format string

6.4.1.9 `get_concurrency_level()`

```
unsigned int utility::get_concurrency_level ( )
```

Gets the concurrency level.

The concurrency level is either the number of logical cores of the system or the value passed by the user with the `-t` flag.

Returns

the number of concurrent threads

6.4.1.10 `idx_to_base()`

```
char utility::idx_to_base (
    int idx )
```

Base id to char base.

Converts the given id to the correspondent nucleotide. The mapping used is {0: 'A', 1: 'T', 2: 'G', 3: 'C'}.

Parameters

<i>idx</i>	integer base id
------------	-----------------

Returns

nucleotide character

Exceptions

<i>std::invalid_argument</i>	when <i>idx</i> < 0 or <i>idx</i> > 3
------------------------------	---------------------------------------

6.4.1.11 is_command_available()

```
bool utility::is_command_available (
    const char * command )
```

Checks if the given command is available through the system shell.

Parameters

<i>command</i>	the name of the command
----------------	-------------------------

Returns

true if the command is available, false otherwise

6.4.1.12 map_alignments()

```
void utility::map_alignments (
    const char * filename,
    AlignmentCollection * collection,
    const unordered_map< string, uint32_t > & contig_name_to_id )
```

Reads and map aligned reads to contigs.

Reads the given SAM file and clusters the alignments around the contig they reference.

Parameters

<i>filename</i>	path to SAM file with alignments
<i>collection</i>	object to be filled by the function call
<i>contig_name_to_id</i>	map with contig names as keys and integer contig ids as values

6.4.1.13 read_fasta()

```
void utility::read_fasta (
    StringSet< CharString > * pids,
    StringSet< Dna5String > * pseqs,
    char * ont_reads_filename )
```

Reads a FASTA file.

Reads sequences data from FASTA file and stores it in two sets: sequences ids and sequences

Parameters

<i>pids</i>	pointer to the set of ids
<i>pseqs</i>	pointer to the set of sequences
<i>ont_reads_filename</i>	[description]

6.4.1.14 read_sam() [1/2]

```
void utility::read_sam (
    BamHeader * pheader,
    vector< BamAlignmentRecord > * precords,
    const char * filename )
```

Read a SAM file.

Reads alignment data from the given SAM file and stores the sequence data in the objects passed as arguments to this function.

Parameters

<i>pheader</i>	pointer to a BAM/SAM file header
<i>precords</i>	pointer to the vector where alignments will be stored
<i>filename</i>	path to the input SAM file

6.4.1.15 read_sam() [2/2]

```
void utility::read_sam (
    BamFileIn * pinput_file,
    BamHeader * pheader,
    vector< BamAlignmentRecord > * precords )
```

6.4.1.16 reverse_complement()

```
string utility::reverse_complement (
    const Dna5String & seq )
```

Method creates reverse complement from string given as parameter.

Parameters

<i>seq</i>	String that needs to be reverse complemented.
------------	---

Returns

Reverse complement of string given as parameter.

6.4.1.17 set_concurrency_level()

```
void utility::set_concurrency_level (
    int threads )
```

Sets the concurrency level.

Set the concurrency level to the given number of threads.

Parameters

<i>threads</i>	the number of parallel threads
----------------	--------------------------------

6.4.1.18 throw_exception()

```
template<typename T >
void utility::throw_exception (
    const char * format,
    ... )
```

Throw an exception.

Throw an exception of the given template type with the provided error message.

Parameters

<i>format</i>	the the format of the error message
...	printf style arguments to fill the format string

Template Parameters

<i>T</i>	the type of exception to be thrown
----------	------------------------------------

6.4.1.19 write_fasta() [1/2]

```
void utility::write_fasta (
    const CharString & id,
    const Dna5String & seq,
    const char * filename )
```

Write a sequence to file.

Writes a single sequence to a FASTA file.

Parameters

<i>id</i>	string ID of the sequence
<i>seq</i>	bases of the sequence
<i>filename</i>	path to the output file

6.4.1.20 write_fasta() [2/2]

```
void utility::write_fasta (
    const StringSet< CharString > & ids,
    const StringSet< Dna5String > & seqs,
    const char * filename )
```

Writes a set of sequences to file.

Writes multiple sequences to a FASTA file. String ids and sequence contents at the same index will be grouped together to form one FASTA entry.

Parameters

<i>ids</i>	collection of the string ids of the sequences
<i>seqs</i>	collection of the bases of the sequences
<i>filename</i>	path to the output file

6.4.2 Variable Documentation

6.4.2.1 command_buffer

```
char utility::command_buffer = { 0 }
```

Buffer to hold shell command strings.

6.4.2.2 error_buffer

```
char utility::error_buffer = { 0 }
```

Buffer to hold a description string when an error occurs.

6.4.2.3 hardware_concurrency

```
unsigned int utility::hardware_concurrency
```

Initial value:

```
= max(  
    1u, std::thread::hardware_concurrency())
```

6.4.2.4 seq_id_buffer

```
char utility::seq_id_buffer = { 0 }
```

Buffer used to build the name of a sequence.

Chapter 7

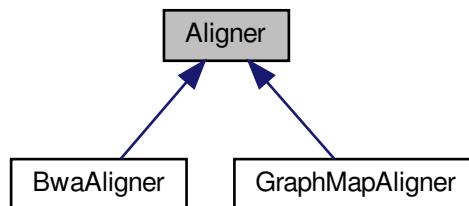
Class Documentation

7.1 Aligner Class Reference

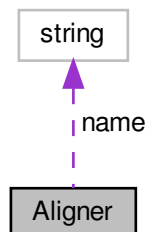
Class representing an abstract aligner.

```
#include <aligner.h>
```

Inheritance diagram for Aligner:



Collaboration diagram for Aligner:



Public Member Functions

- virtual `~Aligner()`=default
The default virtual destructor for this class.
- virtual void `index` (const char *filename)=0
Generate an index for the given genome.
- virtual void `align` (const char *reference_file, const char *reads_file)=0
Align the given reads to a reference genome.
- virtual void `align` (const char *reference_file, const char *reads_file, const char *sam_file, bool only_↔ primary)=0
- virtual void `align` (const char *reference_file, const char *reads_file, const char *sam_file)=0
- virtual void `align` (const CharString &id, const Dna5String &contig, const char *reads_filename)=0
- const std::string & `get_name` () const

Static Public Member Functions

- static const char * `get_tmp_alignment_filename` ()
- static const char * `get_tmp_reference_filename` ()
- static const char * `get_tmp_contig_filename` ()
- static void `init` (bool use_graphmap_aligner, read_type::ReadType read_type)
- static `Aligner` & `get_instance` ()

Protected Member Functions

- `Aligner` (const std::string &name, read_type::ReadType tech_type)
Constructor for `Aligner`.

Protected Attributes

- std::string `name`
The name of the aligner.
- read_type::ReadType `tech_type`
The type of reads that will be used as input.

7.1.1 Detailed Description

Class representing an abstract aligner.

The `Aligner` class is an abstract class that defines the minimum interface that an aligner needs to implement in order to be used in the scaffolding pipeline.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Aligner()

```
Aligner::Aligner (
    const std::string & name,
    read_type::ReadType tech_type ) [inline], [protected]
```

Constructor for `Aligner`.

Parameters

<i>name</i>	the name of the aligner
<i>tech_type</i>	the type of the reads that will be used as input

7.1.2.2 ~Aligner()

```
virtual Aligner::~Aligner ( ) [virtual], [default]
```

The default virtual destructor for this class.

7.1.3 Member Function Documentation

7.1.3.1 align() [1/4]

```
virtual void Aligner::align (
    const char * reference_file,
    const char * reads_file ) [pure virtual]
```

Align the given reads to a reference genome.

Parameters

<i>reference_file</i>	[description]
<i>reads_file</i>	[description]

Implemented in [BwaAligner](#), and [GraphMapAligner](#).

7.1.3.2 align() [2/4]

```
virtual void Aligner::align (
    const char * reference_file,
    const char * reads_file,
    const char * sam_file,
    bool only_primary ) [pure virtual]
```

Implemented in [BwaAligner](#), and [GraphMapAligner](#).

7.1.3.3 align() [3/4]

```
virtual void Aligner::align (
    const char * reference_file,
    const char * reads_file,
    const char * sam_file ) [pure virtual]
```

Implemented in [BwaAligner](#), and [GraphMapAligner](#).

7.1.3.4 align() [4/4]

```
virtual void Aligner::align (
    const CharString & id,
    const Dna5String & contig,
    const char * reads_filename ) [pure virtual]
```

Implemented in [BwaAligner](#), and [GraphMapAligner](#).

7.1.3.5 get_instance()

```
Aligner & Aligner::get_instance ( ) [static]
```

7.1.3.6 get_name()

```
const std::string & Aligner::get_name ( ) const
```

7.1.3.7 get_tmp_alignment_filename()

```
const char * Aligner::get_tmp_alignment_filename ( ) [static]
```

7.1.3.8 get_tmp_contig_filename()

```
const char * Aligner::get_tmp_contig_filename ( ) [static]
```

7.1.3.9 get_tmp_reference_filename()

```
const char * Aligner::get_tmp_reference_filename ( ) [static]
```

7.1.3.10 index()

```
virtual void Aligner::index (
    const char * filename ) [pure virtual]
```

Generate an index for the given genome.

Creates the index that is used during alignment for the given genome. The output files are defined by the invoked external command.

Parameters

<i>filename</i>	path to a genome in FASTA format
-----------------	----------------------------------

Implemented in [BwaAligner](#), and [GraphMapAligner](#).

7.1.3.11 init()

```
void Aligner::init (
    bool use_graphmap_aligner,
    read_type::ReadType read_type ) [static]
```

7.1.4 Member Data Documentation

7.1.4.1 name

```
std::string Aligner::name [protected]
```

The name of the aligner.

7.1.4.2 tech_type

```
read_type::ReadType Aligner::tech_type [protected]
```

The type of reads that will be used as input.

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

- [src/aligners/aligner.h](#)
- [src/aligners/aligner.cpp](#)

7.2 bases::BasesCounter Class Reference

Used for calculating various statistics at specific positions in the contig extension process.

```
#include <bases.h>
```

Public Member Functions

- [BasesCounter](#) ()
BasesCounter class default constructor.
- void [digest_base](#) (char base)
Proccesing a single base.
- void [refresh_stats](#) ()
Refresh all member variables.

Public Attributes

- uint32_t [count](#) [NUM_BASES]
array used to store the number of appearances of each base
- uint32_t [coverage](#)
number of bases at this position, sum(count)
- uint32_t [max_idx](#)
index in the count array of base with most appearances

7.2.1 Detailed Description

Used for calculating various statistics at specific positions in the contig extension process.

The [BasesCounter](#) class is used for storing the frequency of appearances of each base at a specific position. The class also calculates the coverage and the most frequent base at the given position.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 BasesCounter()

```
bases::BasesCounter::BasesCounter ( )
```

[BasesCounter](#) class default constructor.

7.2.3 Member Function Documentation

7.2.3.1 digest_base()

```
void bases::BasesCounter::digest_base (
    char base )
```

Proccesing a single base.

Converts the given base to it's corresponding index in the count array and increments the count at that index by one.

Parameters

<i>base</i>	character base in a DNA sequence
-------------	----------------------------------

7.2.3.2 refresh_stats()

```
void bases::BasesCounter::refresh_stats ( )
```

Refresh all member variables.

The count array is used to compute the coverage and the most frequent base.

7.2.4 Member Data Documentation

7.2.4.1 count

```
uint32_t bases::BasesCounter::count [NUM_BASES]
```

array used to store the number of appearances of each base

7.2.4.2 coverage

```
uint32_t bases::BasesCounter::coverage
```

number of bases at this position, sum(count)

7.2.4.3 max_idx

```
uint32_t bases::BasesCounter::max_idx
```

index in the count array of base with most appearances

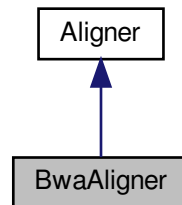
The documentation for this class was generated from the following files:

- [src/bases.h](#)
- [src/bases.cpp](#)

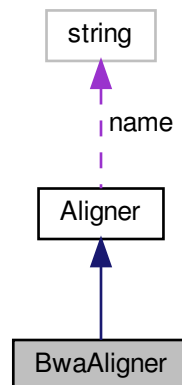
7.3 BwaAligner Class Reference

```
#include <bwa.h>
```

Inheritance diagram for BwaAligner:



Collaboration diagram for BwaAligner:



Public Member Functions

- [BwaAligner](#) ([read_type::ReadType](#) tech_type)
- virtual void [index](#) (const char *filename)
Bwa index command wrapper.
- virtual void [align](#) (const char *reference_file, const char *reads_file)
Bwa mem command wrapper.
- virtual void [align](#) (const char *reference_file, const char *reads_file, const char *sam_file, bool only_primary)
Bwa mem command wrapper.
- virtual void [align](#) (const char *reference_file, const char *reads_file, const char *sam_file)
Bwa mem command wrapper.
- virtual void [align](#) (const CharString &id, const Dna5String &contig, const char *reads_filename)
Align reads from file to contig using bwa.

Additional Inherited Members

7.3.1 Detailed Description

[Aligner](#) namespace. It consists of various bwa tool wrapper functions which make system calls to execute these bwa commands.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 BwaAligner()

```
BwaAligner::BwaAligner (
    read_type::ReadType tech_type ) [inline], [explicit]
```

7.3.3 Member Function Documentation

7.3.3.1 align() [1/4]

```
void BwaAligner::align (
    const char * reference_file,
    const char * reads_file ) [virtual]
```

Bwa mem command wrapper.

Method makes system call to execute bwa mem command. Aligns reads from read file to reference in reference file (index has to be created for this reference before). Alignment results are stored in tmp_alignment_filename and bwa mem command is executed in verbose mode.

Parameters

<i>reference_file</i>	FASTA file with reference sequence(s).
<i>reads_file</i>	FASTA file with reads.

Implements [Aligner](#).

7.3.3.2 align() [2/4]

```
void BwaAligner::align (
    const char * reference_file,
```

```

    const char * reads_file,
    const char * sam_file,
    bool only_primary ) [virtual]

```

Bwa mem command wrapper.

Method makes system call to execute bwa mem command. Aligns reads from read file to reference in reference file (index has to be created for this reference before).

Parameters

<i>reference_file</i>	FASTA file with reference sequence(s).
<i>reads_file</i>	FASTA file with reads.
<i>sam_file</i>	SAM file for storing results of bwa mem command.
<i>only_primary</i>	Flag for executing bwa mem in verbose mode. Verbose mode is turned on if flag is set to false.

Implements [Aligner](#).

7.3.3.3 align() [3/4]

```

void BwaAligner::align (
    const char * reference_file,
    const char * reads_file,
    const char * sam_file ) [virtual]

```

Bwa mem command wrapper.

Method makes system call to execute bwa mem command. Aligns reads from read file to reference in reference file (index has to be created for this reference before). Bwa mem command is executed in verbose mode.

Parameters

<i>reference_file</i>	FASTA file with reference sequence(s).
<i>reads_file</i>	FASTA file with reads.
<i>sam_file</i>	SAM file for storing results of bwa mem command.

Implements [Aligner](#).

7.3.3.4 align() [4/4]

```

void BwaAligner::align (
    const CharString & id,
    const Dna5String & contig,
    const char * reads_filename ) [virtual]

```

Align reads from file to contig using bwa.

Method writes contig sequence to tmp_contig_filename, creates index using bwa index command and align reads to contig using bwa mem command.

Deprecated

Parameters

<i>id</i>	Contig id.
<i>contig</i>	Contig sequence.
<i>reads_filename</i>	FASTA file with reads.

Implements [Aligner](#).

7.3.3.5 index()

```
void BwaAligner::index (
    const char * filename ) [virtual]
```

Bwa index command wrapper.

Method makes system call to execute bwa index command. Command creates index for sequences stored in file.

Parameters

<i>filename</i>	FASTA file with sequences for creating index.
-----------------	---

Implements [Aligner](#).

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

- [src/aligners/bwa.h](#)
- [src/aligners/bwa.cpp](#)

7.4 Connector Class Reference

[Connector](#) class.

```
#include <connector.h>
```

Public Member Functions

- [Connector](#) (const vector< [Contig](#) *> &contigs)
[Connector](#) class constructor.
- [~Connector](#) ()
[Connector](#) class destructor.
- void [connect_contigs](#) (bool trim_circular_genome)
Method connects contigs passed as input into scaffolds if this is possible.
- const vector< [Scaffold](#) * > & [get_scaffolds](#) ()
Getter for scaffolds.
- void [dump_scaffolds](#) (const char *output_file)
Output scaffolds to file.

7.4.1 Detailed Description

[Connector](#) class.

Class provides functionality for connecting extended contigs into scaffolds if they mutually overlap.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 Connector()

```
Connector::Connector (
    const vector< Contig *> & contigs ) [explicit]
```

[Connector](#) class constructor.

Constructor initializes [Connector](#) class with contigs which are afterwards merged into scaffolds.

Parameters

<i>contigs</i>	Vector of contigs.
----------------	--------------------

7.4.2.2 ~Connector()

```
Connector::~~Connector ( )
```

[Connector](#) class destructor.

7.4.3 Member Function Documentation

7.4.3.1 connect_contigs()

```
void Connector::connect_contigs (
    bool trim_circular_genome )
```

Method connects contigs passed as input into scaffolds if this is possible.

Anchors are created from each contigs. For each contig, overlap between anchors and this contig are found (if any). This contig is afterwards merged with contig that anchor belongs to. Procedure is repeated until all contigs are processed.

Parameters

<i>trim_circular_genome</i>	flag to enable/disable trimming excessive bases from circular genomes
-----------------------------	---

7.4.3.2 dump_scaffolds()

```
void Connector::dump_scaffolds (
    const char * output_file )
```

Output scaffolds to file.

Scaffolds are outputed in FASTA format file genome.fasta

Parameters

<i>output_file</i>	path to the output file
--------------------	-------------------------

7.4.3.3 get_scaffolds()

```
const vector<Scaffold*>& Connector::get_scaffolds ( ) [inline]
```

Getter for scaffolds.

Returns

Vector of scaffolds.

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

- src/[connector.h](#)
- src/[connector.cpp](#)

7.5 Contig Class Reference

[Contig](#) class represent contig sequences.

```
#include <contig.h>
```

Public Member Functions

- [Contig](#) ()
Contig class default constructor.
- [Contig](#) (Dna5String &seq, int total_ext_left, int total_ext_right)
Contig class constructor.
- [Contig](#) (const Dna5String &contig_seq, string &left_extension, string &right_extension)
Contig class constructor.
- Dna5String & [seq](#) ()
Getter for extended contig sequence.
- CharString & [id](#) ()
Getter for contig id.
- int [total_len](#) ()
Getter for total length of extended contig sequence.
- int [total_ext_left](#) ()
Getter for length of left extension.
- int [total_ext_right](#) ()
Getter for length of right extension.
- int [right_ext_pos](#) ()
Getter for right extension start position in extended contig sequence.
- string & [ext_left](#) ()
Getter for left extension.
- string & [ext_right](#) ()
Getter for right extension.
- CharString & [left_id](#) ()
Getter for left extension id.
- CharString & [right_id](#) ()
Getter for right extension id.
- void [set_id](#) (const CharString &id)
Setter for contig id.
- void [reverse_complement](#) ()
Method reverse complements this contig.
- bool [operator==](#) (const [Contig](#) &other) const
Operator == overload.
- bool [operator!=](#) (const [Contig](#) &other) const
Operator != overload.

Static Public Member Functions

- static void [dump_anchors](#) (const vector< [Contig](#) *> &contigs, const char *anchors_file)
Method creates left and right anchor from contigs and dumps them to file.

7.5.1 Detailed Description

[Contig](#) class represent contig sequences.

[Contig](#) class is wrapper for original contigs. It provides functionality for accessing extensions of contig found in extension process. It also provides functionality for accessing contig anchors - subsequences of contig used in contigs merge process.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Contig() [1/3]

```
Contig::Contig ( )
```

[Contig](#) class default constructor.

7.5.2.2 Contig() [2/3]

```
Contig::Contig (
    Dna5String & seq,
    int total_ext_left,
    int total_ext_right )
```

[Contig](#) class constructor.

Constructor used for creating [Contig](#) object from already extended contig and lengths of right and left extensions.

Parameters

<i>seq</i>	Extended contig sequence.
<i>total_ext_left</i>	Length of left extension.
<i>total_ext_right</i>	Length of right extension.

7.5.2.3 Contig() [3/3]

```
Contig::Contig (
    const Dna5String & contig_seq,
    string & left_extension,
    string & right_extension )
```

[Contig](#) class constructor.

Constructor used for creating [Contig](#) object from original contig sequence and sequences of left and right extensions.

Parameters

<i>contig_seq</i>	Original contig sequence.
<i>left_extension</i>	Left extension sequence.
<i>right_extension</i>	Right extension sequence.

7.5.3 Member Function Documentation

7.5.3.1 dump_anchors()

```
static void Contig::dump_anchors (
    const vector< Contig *> & contigs,
    const char * anchors_file ) [inline], [static]
```

Method creates left and right anchor from contigs and dumps them to file.

Parameters

<i>contigs</i>	contigs used for creation of anchors
<i>anchors_file</i>	output file to dump the anchors

7.5.3.2 ext_left()

```
string& Contig::ext_left ( ) [inline]
```

Getter for left extension.

Returns

[Contig](#) left extension.

7.5.3.3 ext_right()

```
string& Contig::ext_right ( ) [inline]
```

Getter for right extension.

Returns

[Contig](#) right extension.

7.5.3.4 id()

```
CharString& Contig::id ( ) [inline]
```

Getter for contig id.

Returns

[Contig](#) id.

7.5.3.5 left_id()

```
CharString& Contig::left_id ( ) [inline]
```

Getter for left extension id.

Returns

Left extension id.

7.5.3.6 operator!=()

```
bool Contig::operator!= (
    const Contig & other ) const [inline]
```

Operator != overload.

Returns

True if contigs are different, false otherwise.

7.5.3.7 operator==()

```
bool Contig::operator== (
    const Contig & other ) const [inline]
```

Operator == overload.

Returns

True if contigs are equal, false otherwise.

7.5.3.8 reverse_complement()

```
void Contig::reverse_complement ( )
```

Method reverse complements this contig.

Extended sequence is reversed and complemented. Left and right extensions data is updated from this reverse complemented sequence.

7.5.3.9 right_ext_pos()

```
int Contig::right_ext_pos ( ) [inline]
```

Getter for right extension start position in extended contig sequence.

Returns

Start index of right extension in extended contig sequence.

7.5.3.10 right_id()

```
CharString& Contig::right_id ( ) [inline]
```

Getter for right extension id.

Returns

Right extension id.

7.5.3.11 seq()

```
Dna5String& Contig::seq ( ) [inline]
```

Getter for extended contig sequence.

Returns

[Contig](#) extended sequence.

7.5.3.12 set_id()

```
void Contig::set_id (
    const CharString & id )
```

Setter for contig id.

Parameters

<i>id</i>	Contig id.
-----------	----------------------------

7.5.3.13 total_ext_left()

```
int Contig::total_ext_left ( ) [inline]
```

Getter for length of left extension.

Returns

Length of left extension

7.5.3.14 total_ext_right()

```
int Contig::total_ext_right ( ) [inline]
```

Getter for length of right extension.

Returns

Length of right extension.

7.5.3.15 total_len()

```
int Contig::total_len ( ) [inline]
```

Getter for total length of extended contig sequence.

Returns

Length of extended contig sequence.

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

- [src/contig.h](#)
- [src/contig.cpp](#)

7.6 Extension Class Reference

[Extension](#) class.

```
#include <extension.h>
```

Public Member Functions

- [Extension](#) (uint32_t [read_id](#), const string &[seq](#), bool drop)
[Extension](#) class constructor.
- uint32_t [read_id](#) ()
Getter for read Id.
- const string & [seq](#) ()
Getter for possible extension sequence.
- uint32_t [curr_pos](#) ()
Getter for current position in extension durring extension process.
- void [do_operation](#) (const [Operation](#) &op)
Local realignment operation executor.

Public Attributes

- bool [is_dropped](#)
True if the extension is dropped, false otherwise.

7.6.1 Detailed Description

[Extension](#) class.

[Extension](#) class is used as representation of possible extension reads of contig. It provides functionality for local realignment method used in process.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 [Extension](#)()

```
Extension::Extension (  
    uint32_t read_id,  
    const string & seq,  
    bool drop )
```

[Extension](#) class constructor.

Parameters

<i>read↵ _id</i>	Id of read that is possible extension.
<i>seq</i>	Subsequence of read sequence that is possible extension.
<i>drop</i>	Bool value that representes if this read is dropped.

7.6.3 Member Function Documentation

7.6.3.1 curr_pos()

```
uint32_t Extension::curr_pos ( ) [inline]
```

Getter for current position in extension durring extension process.

Returns

Index of current position in extension sequence.

7.6.3.2 do_operation()

```
void Extension::do_operation (
    const Operation & op )
```

Local realignment operation executor.

Depending on operation current position in extension sequence is moved ahead by 1 or 2 or remains unchanged.

Parameters

<i>op</i>	Alignment operation.
-----------	----------------------

7.6.3.3 read_id()

```
uint32_t Extension::read_id ( ) [inline]
```

Getter for read Id.

Returns

Read Id.

7.6.3.4 seq()

```
const string& Extension::seq ( ) [inline]
```

Getter for possible extension sequence.

Returns

[Extension](#) sequence.

7.6.4 Member Data Documentation

7.6.4.1 is_dropped

```
bool Extension::is_dropped
```

True if the extension is dropped, false otherwise.

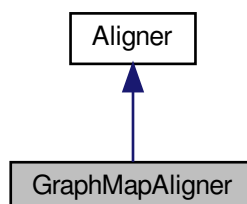
The documentation for this class was generated from the following files:

- [src/extension.h](#)
- [src/extension.cpp](#)

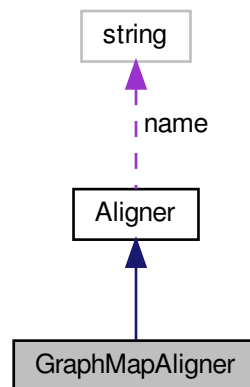
7.7 GraphMapAligner Class Reference

```
#include <graphmap.h>
```

Inheritance diagram for GraphMapAligner:



Collaboration diagram for GraphMapAligner:



Public Member Functions

- [GraphMapAligner](#) ([read_type::ReadType](#) tech_type)
- virtual [~GraphMapAligner](#) ()=default
- virtual void [index](#) (const char *filename)
Generate an index for the given genome.
- virtual void [align](#) (const char *reference_file, const char *reads_file)
Align the given reads to a reference genome.
- virtual void [align](#) (const char *reference_file, const char *reads_file, const char *sam_file, bool only_primary)
- virtual void [align](#) (const char *reference_file, const char *reads_file, const char *sam_file)
- virtual void [align](#) (const CharString &id, const Dna5String &contig, const char *reads_filename)

Additional Inherited Members

7.7.1 Constructor & Destructor Documentation

7.7.1.1 GraphMapAligner()

```

GraphMapAligner::GraphMapAligner (
    read\_type::ReadType tech_type ) [inline], [explicit]
  
```

7.7.1.2 ~GraphMapAligner()

```

virtual GraphMapAligner::~~GraphMapAligner ( ) [virtual], [default]
  
```

7.7.2 Member Function Documentation

7.7.2.1 `align()` [1/4]

```
void GraphMapAligner::align (
    const char * reference_file,
    const char * reads_file ) [virtual]
```

Align the given reads to a reference genome.

Parameters

<i>reference_file</i>	[description]
<i>reads_file</i>	[description]

Implements [Aligner](#).

7.7.2.2 `align()` [2/4]

```
void GraphMapAligner::align (
    const char * reference_file,
    const char * reads_file,
    const char * sam_file,
    bool only_primary ) [virtual]
```

Implements [Aligner](#).

7.7.2.3 `align()` [3/4]

```
void GraphMapAligner::align (
    const char * reference_file,
    const char * reads_file,
    const char * sam_file ) [virtual]
```

Implements [Aligner](#).

7.7.2.4 `align()` [4/4]

```
void GraphMapAligner::align (
    const CharString & id,
    const Dna5String & contig,
    const char * reads_filename ) [virtual]
```

Implements [Aligner](#).

7.7.2.5 index()

```
void GraphMapAligner::index (
    const char * filename ) [virtual]
```

Generate an index for the given genome.

Creates the index that is used during alignment for the given genome. The output files are defined by the invoked external command.

Parameters

<i>filename</i>	path to a genome in FASTA format
-----------------	----------------------------------

Implements [Aligner](#).

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

- [src/aligners/graphmap.h](#)
- [src/aligners/graphmap.cpp](#)

7.8 Scaffold Class Reference

[Scaffold](#) class is representation of scaffold structure.

```
#include <scaffold.h>
```

Public Member Functions

- [Scaffold](#) ([Contig](#) *first_contig)
Scaffold constructor.
- void [add_contig](#) ([Contig](#) *contig, int last_end, int this_start)
Method adds next contig to scaffold structure.
- bool [contains](#) (const string &id)
Checks if scaffold contains contig with given id.
- [Dna5String](#) [get_combined_sequence](#) ()
Method creates unified scaffold sequence.
- [Contig](#) * [first_contig](#) ()
Getter for first contig in scaffold.
- [Contig](#) * [last_contig](#) ()
Getter for last contig in scaffold.
- const vector< [Contig](#) * > & [get_contigs](#) ()
Getter for all contigs in scaffold.
- int [num_contigs](#) ()
Getter for number of contigs in scaffold.
- void [merge](#) ([Scaffold](#) *scaffold)
Merge this scaffold with scaffold given as parameter.
- void [trim](#) (int left_start_pos, int right_end_pos)
Trim scaffold if it is circular.

7.8.1 Detailed Description

[Scaffold](#) class is representation of scaffold structure.

[Scaffold](#) class is used as representation of scaffolds. [Scaffold](#) consists of multiple Contigs and memorizes contributions of each contig into scaffold sequence.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Scaffold()

```
Scaffold::Scaffold (
    Contig * first_contig ) [explicit]
```

[Scaffold](#) constructor.

Constructor creates scaffold from initial contig.

Parameters

<i>first_contig</i>	First contig scaffold consists of.
---------------------	------------------------------------

7.8.3 Member Function Documentation

7.8.3.1 add_contig()

```
void Scaffold::add_contig (
    Contig * contig,
    int last_end,
    int this_start )
```

Method adds next contig to scaffold structure.

[Contig](#) is added as last contig in a scaffold. Because of overlaps between previously last contig and the one being added contributions of both contigs to scaffold sequence have to be updated.

Parameters

<i>contig</i>	Contig to be added.
<i>last_end</i>	End contribution index of previously last contig.
<i>this_start</i>	Start contribution index of contig being added.

7.8.3.2 contains()

```
bool Scaffold::contains (
    const string & id ) [inline]
```

Checks if scaffold contains contig with given Id.

Parameters

<i>id</i>	Contig Id.
-----------	------------

Returns

True if contains, false otherwise.

7.8.3.3 first_contig()

```
Contig* Scaffold::first_contig ( ) [inline]
```

Getter for first contig in scaffold.

Returns

First contig in scaffold.

7.8.3.4 get_combined_sequence()

```
Dna5String Scaffold::get_combined_sequence ( )
```

Method creates unified scaffold sequence.

One scaffold sequence is generated from multiple contig sequences. Depending on contributions of each contig, subsequences from contig sequences are extracted and connected with each other.

Returns

Scaffold sequence.

7.8.3.5 get_contigs()

```
const vector<Contig*>& Scaffold::get_contigs ( ) [inline]
```

Getter for all contigs in scaffold.

Returns

Contigs scaffold consists of.

7.8.3.6 last_contig()

```
Contig* Scaffold::last_contig ( ) [inline]
```

Getter for last contig in scaffold.

Returns

Last contig in scaffold.

7.8.3.7 merge()

```
void Scaffold::merge (
    Scaffold * scaffold )
```

Merge this scaffold with scaffold given as parameter.

All contigs from scaffold given as parameter are inserted at the end of vector of contigs of this scaffold. Also, contigs contributions are updated.

Parameters

<i>scaffold</i>	Scaffold to merge into this one.
-----------------	--

7.8.3.8 num_contigs()

```
int Scaffold::num_contigs ( ) [inline]
```

Getter for number of contigs in scaffold.

Returns

Number of contigs in scaffold.

7.8.3.9 trim()

```
void Scaffold::trim (
    int left_start_pos,
    int right_end_pos )
```

Trim scaffold if it is circular.

Only contributions of first and last contig in scaffold have to be modified accordingly to parameters passed.

Parameters

<i>left_start_pos</i>	Start index in sequence of first contig.
<i>right_end_pos</i>	End index in sequence of last contig.

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

- [src/scaffold.h](#)
- [src/scaffold.cpp](#)

Chapter 8

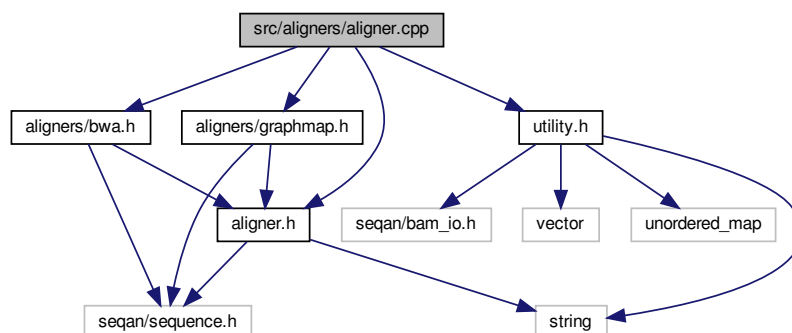
File Documentation

8.1 src/aligners/aligner.cpp File Reference

Implementation file for the abstract [Aligner](#) class.

```
#include "aligners/bwa.h"
#include "aligners/graphmap.h"
#include "aligner.h"
#include "utility.h"
```

Include dependency graph for aligner.cpp:



8.1.1 Detailed Description

Implementation file for the abstract [Aligner](#) class.

Copyright

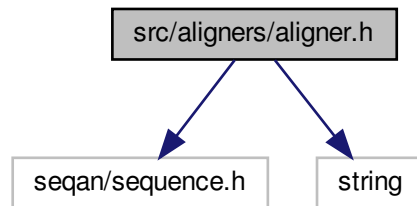
Marko Culinovic marko.culinovic@gmail.com

Luka Sterbic luka.sterbic@gmail.com

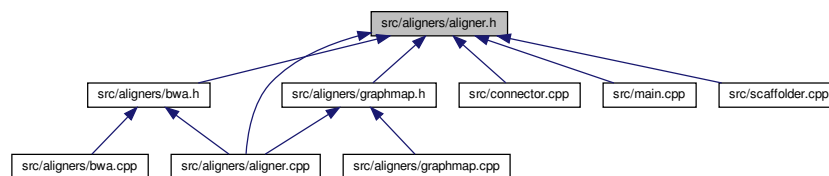
8.2 src/aligners/aligner.h File Reference

Declaration of an abstract aligner class.

```
#include <seqan/sequence.h>
#include <string>
Include dependency graph for aligner.h:
```



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



Classes

- class [Aligner](#)
Class representing an abstract aligner.

Namespaces

- [read_type](#)

Enumerations

- enum [read_type::ReadType](#) { [read_type::PacBio](#), [read_type::ONT](#) }
Enum used to distinguish read types.

Functions

- ReadType [read_type::string_to_read_type](#) (const char *read_type)
Converts the given string to a ReadType enumerator object.

8.2.1 Detailed Description

Declaration of an abstract aligner class.

Copyright

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

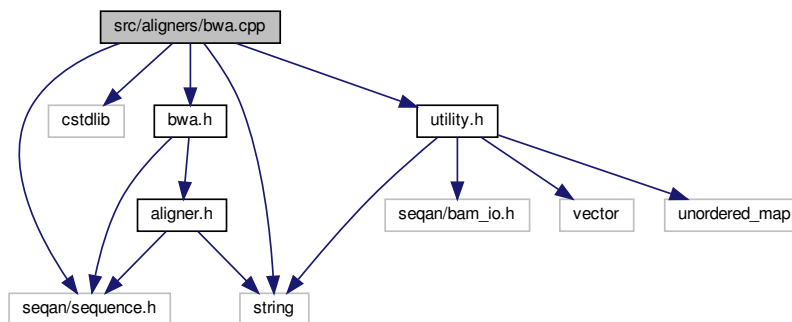
Contains the minimum interface that an aligner needs to implement in order to be usable by eagler.

8.3 src/aligners/bwa.cpp File Reference

Implementation file for aligner namespace.

```
#include <seqan/sequence.h>
#include <cstdlib>
#include <string>
#include "bwa.h"
#include "utility.h"
```

Include dependency graph for bwa.cpp:



8.3.1 Detailed Description

Implementation file for aligner namespace.

Copyright

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

Implementation file for aligner namespace. It consists of various bwa tool wrapper functions which make system calls to execute these bwa commands.

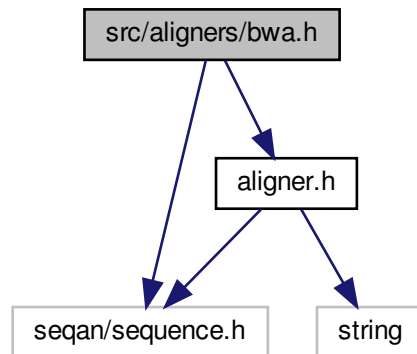
8.4 src/aligners/bwa.h File Reference

Header file for aligner namespace.

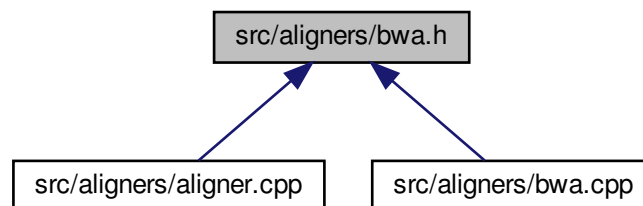
```
#include <seqan/sequence.h>
```

```
#include "aligner.h"
```

Include dependency graph for bwa.h:



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



Classes

- class [BwaAligner](#)

8.4.1 Detailed Description

Header file for aligner namespace.

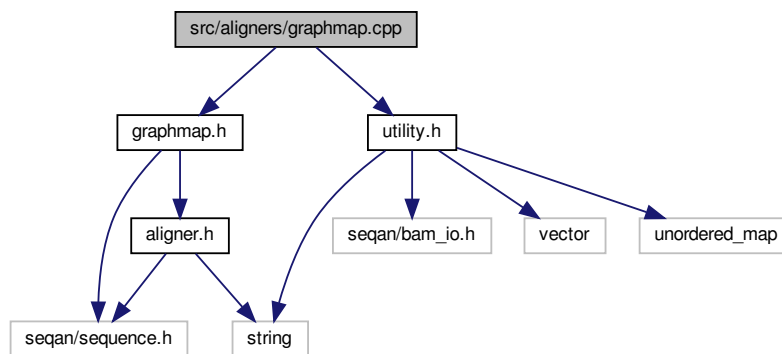
Copyright

Marko Culinovic marko.culinovic@gmail.com
 Luka Sterbic luka.sterbic@gmail.com

Header file for aligner namespace. It consists of various bwa tool wrapper functions which make system calls to execute these bwa commands.

8.5 src/aligners/graphmap.cpp File Reference

```
#include "graphmap.h"
#include "utility.h"
Include dependency graph for graphmap.cpp:
```



8.5.1 Detailed Description

Copyright

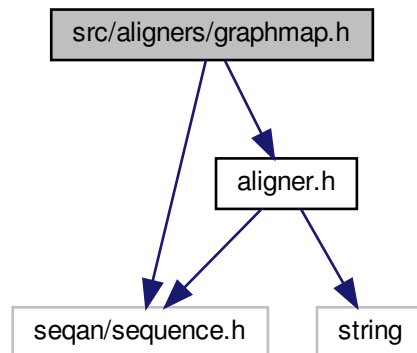
Marko Culinovic marko.culinovic@gmail.com
 Luka Sterbic luka.sterbic@gmail.com

8.6 src/aligners/graphmap.h File Reference

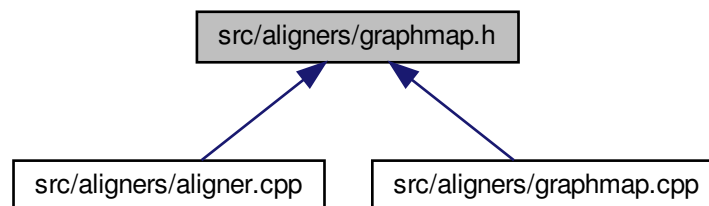
Declaration of the GraphMap class.

```
#include <seqan/sequence.h>
#include "aligner.h"
```

Include dependency graph for graphmap.h:



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



Classes

- class [GraphMapAligner](#)

8.6.1 Detailed Description

Declaration of the GraphMap class.

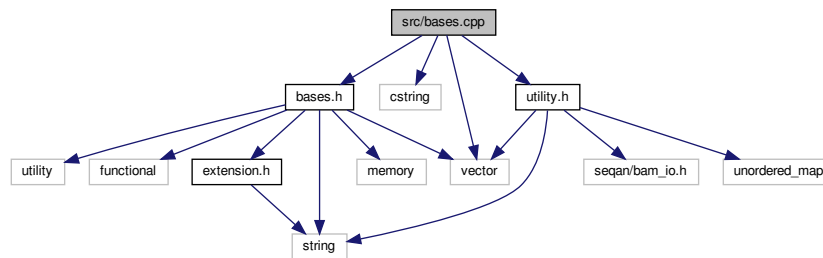
Copyright

Marko Culinovic marko.culinovic@fer.hr
Luka Sterbic luka.sterbic@fer.hr

8.7 src/bases.cpp File Reference

Implementation file for the BasesCounter class and associated factory functions.

```
#include <vector>
#include <cstring>
#include "bases.h"
#include "utility.h"
Include dependency graph for bases.cpp:
```



Namespaces

- [bases](#)

Namespace for [BasesCounter](#) class and supporting functions.

Functions

- BasesCounter [bases::count_bases](#) (const vector< shared_ptr< [Extension](#) >> &extensions, [bool_predicate](#) is_read_eligible, int offset)
Count bases at a specific position in the given extensions.
- BasesCounter [bases::count_bases](#) (const vector< shared_ptr< [Extension](#) >> &extensions)
Count bases at specific position in extensions.

8.7.1 Detailed Description

Implementation file for the BasesCounter class and associated factory functions.

Copyright

Copyright 2015 Marko Culinovic, Luka Sterbic

Author

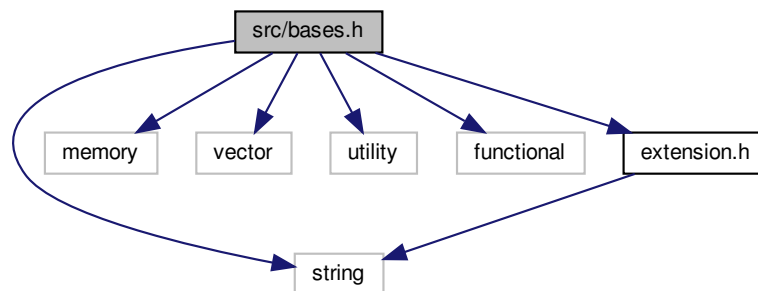
Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

Implementation file for the BasesCounter class and supporting associated factory functions. It is used for calculating various statistics at specific positions in the contig extension process.

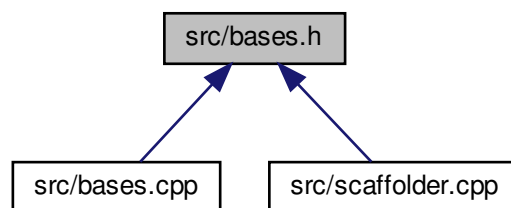
8.8 src/bases.h File Reference

Header file for BasesCounter class and various functions which calculate statistics over a sequence of bases.

```
#include <string>
#include <memory>
#include <vector>
#include <utility>
#include <functional>
#include "extension.h"
Include dependency graph for bases.h:
```



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



Classes

- class `bases::BasesCounter`
Used for calculating various statistics at specific positions in the contig extension process.

Namespaces

- `bases`
Namespace for `BasesCounter` class and supporting functions.

Macros

- `#define NUM_BASES 4`
The number of different bases in a DNA sequence.

Typedefs

- `typedef std::function< bool(char)> bool_predicate`
Type used to represent a boolean function over a character.

Functions

- BasesCounter `bases::count_bases` (const vector< shared_ptr< Extension >> &extensions, bool_predicate is_read_eligible, int offset)
Count bases at a specific position in the given extensions.
- BasesCounter `bases::count_bases` (const vector< shared_ptr< Extension >> &extensions)
Count bases at specific position in extensions.

8.8.1 Detailed Description

Header file for BasesCounter class and various functions which calculate statistics over a sequence of bases.

Copyright

Copyright 2015 Marko Culinovic, Luka Sterbic

Author

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

Header file for BasesCounter class and supporting functions. It is used for calculating various statistics at specific positions in the contig extension process.

8.8.2 Macro Definition Documentation

8.8.2.1 NUM_BASES

```
#define NUM_BASES 4
```

The number of different bases in a DNA sequence.

8.8.3 Typedef Documentation

8.8.3.1 bool_predicate

```
typedef std::function<bool(char)> bool_predicate
```

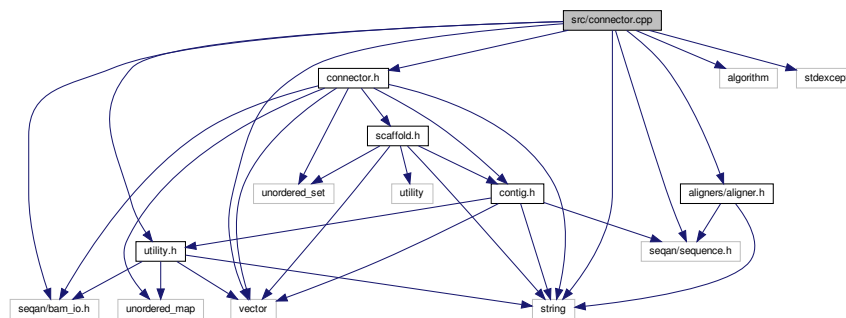
Type used to represent a boolean function over a character.

8.9 src/connector.cpp File Reference

Implementation file for [Connector](#) class.

```
#include <seqan/bam_io.h>
#include <seqan/sequence.h>
#include <vector>
#include <algorithm>
#include <string>
#include <stdexcept>
#include "aligners/aligner.h"
#include "connector.h"
#include "utility.h"
```

Include dependency graph for connector.cpp:



8.9.1 Detailed Description

Implementation file for [Connector](#) class.

Copyright

Marko Culinovic marko.culinovic@gmail.com
 Luka Sterbic luka.sterbic@gmail.com

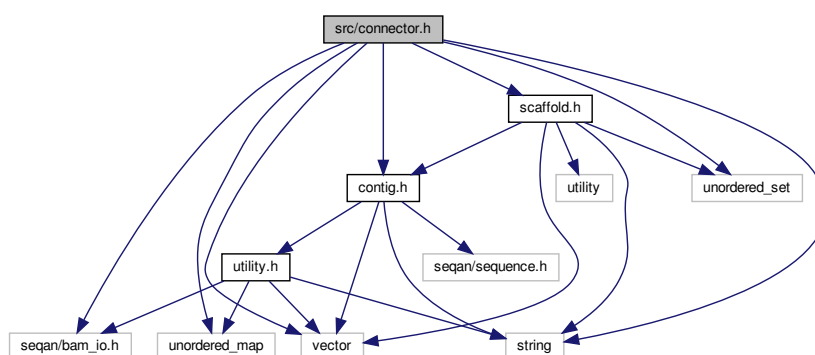
Implementation file for [Connector](#) class. Class provides functionality for connecting extended contigs into scaffolds if they mutually overlap.

8.10 src/connector.h File Reference

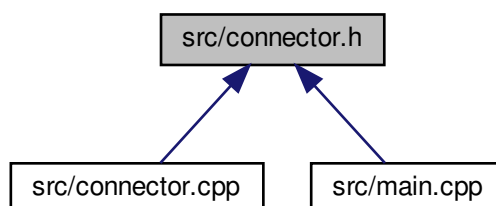
Header file for [Connector](#) class.

```
#include <seqan/bam_io.h>
#include <vector>
#include <unordered_set>
#include <unordered_map>
#include <string>
#include "contig.h"
#include "scaffold.h"
```

Include dependency graph for connector.h:



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



Classes

- class [Connector](#)
Connector class.

Macros

- `#define SECONDARY_LINE 0x900`
SAM file flag denoting a non-primary alignment.
- `#define UNMAPPED 0x4`
SAM file flag denoting an unmapped query.
- `#define COMPLEMENT 0x10`
[brief description] SAM file flag denoting a query from the complement strand of the reference genome
- `#define ANCHOR_THRESHOLD 0.66`
Minimum percentage of anchor sequence that must extend the end of the current scaffold to extend it.

8.10.1 Detailed Description

Header file for `Connector` class.

Copyright

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

Header file for `Connector` class. Class provides functionality for connecting extended contigs into scaffolds if they mutually overlap.

8.10.2 Macro Definition Documentation

8.10.2.1 ANCHOR_THRESHOLD

```
#define ANCHOR_THRESHOLD 0.66
```

Minimum percentage of anchor sequence that must extend the end of the current scaffold to extend it.

8.10.2.2 COMPLEMENT

```
#define COMPLEMENT 0x10
```

[brief description] SAM file flag denoting a query from the complement strand of the reference genome

8.10.2.3 SECONDARY_LINE

```
#define SECONDARY_LINE 0x900
```

SAM file flag denoting a non-primary alignment.

8.10.2.4 UNMAPPED

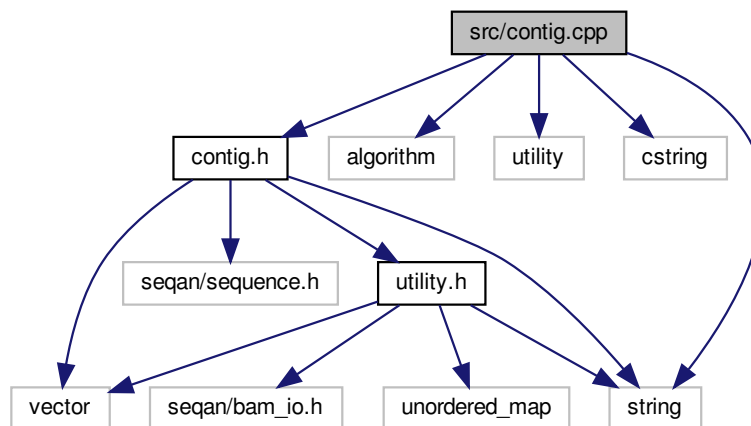
```
#define UNMAPPED 0x4
```

SAM file flag denoting an unmapped query.

8.11 src/contig.cpp File Reference

Implementation file for [Contig](#) class.

```
#include <string>
#include <algorithm>
#include <utility>
#include <cstring>
#include "contig.h"
Include dependency graph for contig.cpp:
```



8.11.1 Detailed Description

Implementation file for [Contig](#) class.

Copyright

Marko Culinovic marko.culinovic@gmail.com
 Luka Sterbic luka.sterbic@gmail.com

Implementation file for [Contig](#) class. [Contig](#) class is wrapper for original contigs. It provides functionality for accessing extensions of contig found in extension process. It also provides functionality for accessing contig anchors - subsequences of contig used in contigs merge process.

8.12.1 Detailed Description

Header file for [Contig](#) class.

Author

Marko Culinovic marko.culinovic@gmail.com

Luka Sterbic luka.sterbic@gmail.com

Header file for [Contig](#) class. [Contig](#) class is wrapper for original contigs. It provides functionality for accessing extensions of contig found in extension process. It also provides functionality for accessing contig anchors - subsequences of contig used in contigs merge process.

8.12.2 Macro Definition Documentation

8.12.2.1 ANCHOR_LEN

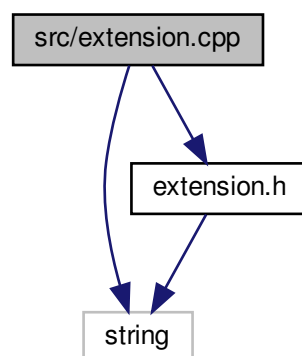
```
#define ANCHOR_LEN 10000
```

8.13 src/extension.cpp File Reference

Implementation file for [Extension](#) class.

```
#include <string>  
#include "extension.h"
```

Include dependency graph for extension.cpp:



8.13.1 Detailed Description

Implementation file for [Extension](#) class.

Copyright

Marko Culinovic marko.culinovic@gmail.com

Luka Sterbic luka.sterbic@gmail.com

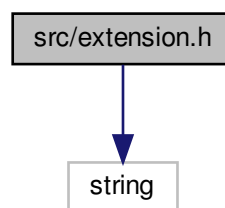
Implementation file for [Extension](#) class. It is used as representation of possible extension reads of contig. It provides functionality for local realignment method used in contig extension process.

8.14 src/extension.h File Reference

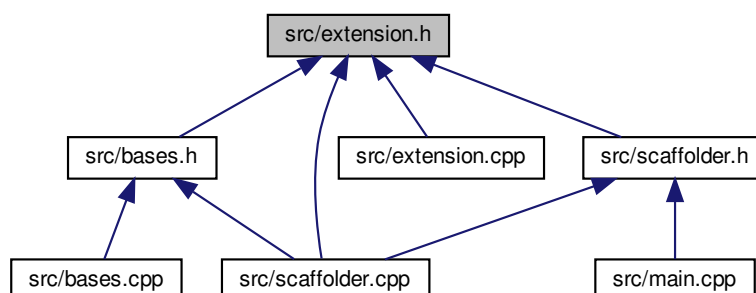
Header file for [Extension](#) class.

```
#include <string>
```

Include dependency graph for extension.h:



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



Classes

- class [Extension](#)
Extension class.

Enumerations

- enum [Operation](#) { [match](#) = 1, [mismatch](#) = 1, [insertion_1](#) = 2, [deletion_1](#) = 0 }

8.14.1 Detailed Description

Header file for [Extension](#) class.

Author

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

Header file for [Extension](#) class. It is used as representation of possible extension reads of contig. It provides functionality for local realignment method used in contig extension process.

8.14.2 Enumeration Type Documentation

8.14.2.1 Operation

enum [Operation](#)

Enum Operation is used for handling moves in local alignment.

Enumerator

match	
mismatch	
insertion↔ _1	
deletion_1	

8.15 src/main.cpp File Reference

Entry point of program and main program of project.

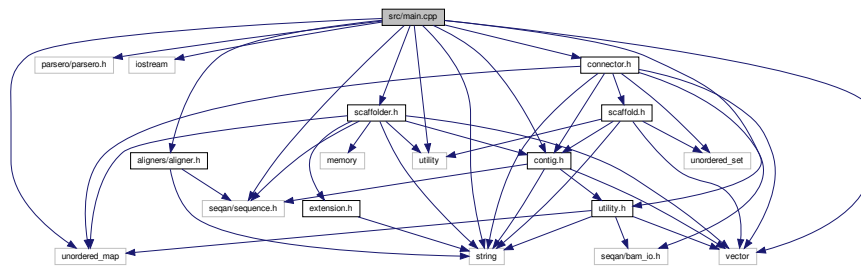
```
#include <seqan/sequence.h>
#include <parsero/parsero.h>
```

```

#include <iostream>
#include <unordered_map>
#include <string>
#include <vector>
#include <utility>
#include "aligners/aligner.h"
#include "utility.h"
#include "scaffolder.h"
#include "contig.h"
#include "connector.h"

```

Include dependency graph for main.cpp:



Macros

- `#define VERSION` ("v1.0.1")
- `#define RELEASE_DATE` (string(__DATE__) + string(" at ") + string(__TIME__))
- `#define PATH_BUFFER_SIZE` 256

Functions

- void `set_output_paths` (char *output_argument)
- void `setup_cmd_interface` (int argc, char **argv)
- int `main` (int argc, char **argv)

Variables

- char * `draft_genome_filename` = nullptr
- char * `reads_filename` = nullptr
- char `tmp_dirname` [PATH_BUFFER_SIZE] = "tmp"
- char `contigs_filename` [PATH_BUFFER_SIZE] = { 0 }
- char `extensions_filename` [PATH_BUFFER_SIZE] = { 0 }
- char `scaffolds_filename` [PATH_BUFFER_SIZE] = { 0 }
- bool `use_POA_consensus` = false
- bool `use_graphmap_aligner` = false
- bool `trim_circular_genome` = true
- `read_type::ReadType` `use_tech_type` = `read_type::PacBio`

8.15.1 Detailed Description

Entry point of program and main program of project.

Copyright

Marko Culinovic marko.culinovic@gmail.com

Luka Sterbic luka.sterbic@gmail.com

8.15.2 Macro Definition Documentation

8.15.2.1 PATH_BUFFER_SIZE

```
#define PATH_BUFFER_SIZE 256
```

8.15.2.2 RELEASE_DATE

```
#define RELEASE_DATE (string(__DATE__) + string(" at ") + string(__TIME__))
```

8.15.2.3 VERSION

```
#define VERSION ("v1.0.1")
```

8.15.3 Function Documentation

8.15.3.1 main()

```
int main (  
    int argc,  
    char ** argv )
```

8.15.3.2 set_output_paths()

```
void set_output_paths (  
    char * output_argument )
```

8.15.3.3 setup_cmd_interface()

```
void setup_cmd_interface (
    int argc,
    char ** argv )
```

8.15.4 Variable Documentation

8.15.4.1 contigs_filename

```
char contigs_filename[PATH_BUFFER_SIZE] = { 0 }
```

8.15.4.2 draft_genome_filename

```
char* draft_genome_filename = nullptr
```

8.15.4.3 extensions_filename

```
char extensions_filename[PATH_BUFFER_SIZE] = { 0 }
```

8.15.4.4 reads_filename

```
char* reads_filename = nullptr
```

8.15.4.5 scaffolds_filename

```
char scaffolds_filename[PATH_BUFFER_SIZE] = { 0 }
```

8.15.4.6 tmp_dirname

```
char tmp_dirname[PATH_BUFFER_SIZE] = "tmp"
```

8.15.4.7 trim_circular_genome

```
bool trim_circular_genome = true
```

8.15.4.8 use_graphmap_aligner

```
bool use_graphmap_aligner = false
```

8.15.4.9 use_POA_consensus

```
bool use_POA_consensus = false
```

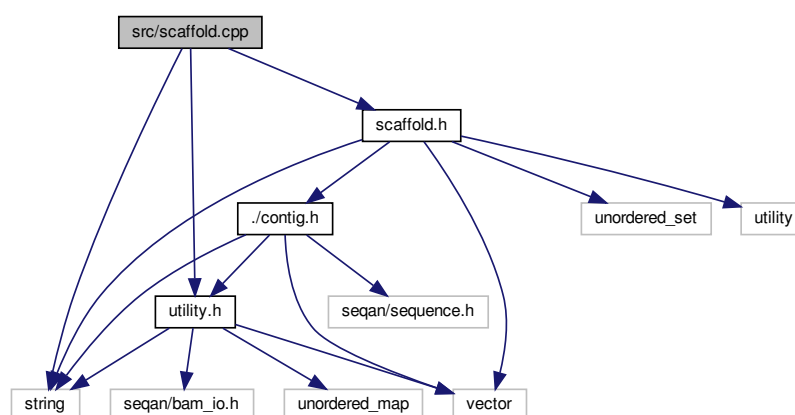
8.15.4.10 use_tech_type

```
read_type::ReadType use_tech_type = read_type::PacBio
```

8.16 src/scaffold.cpp File Reference

Implementation file for [Scaffold](#) class.

```
#include <string>
#include "scaffold.h"
#include "utility.h"
Include dependency graph for scaffold.cpp:
```



8.16.1 Detailed Description

Implementation file for [Scaffold](#) class.

Copyright

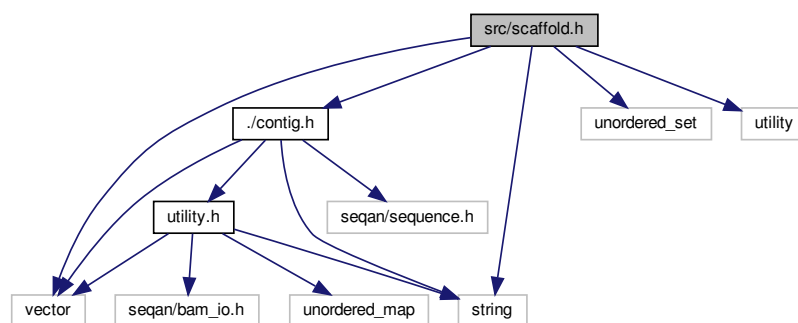
Marko Culinovic marko.culinovic@gmail.com
 Luka Sterbic luka.sterbic@gmail.com

Implementation file for [Scaffold](#) class. It is used as representation of scaffolds. [Scaffold](#) consists of multiple Contigs and memorizes contributions of each contig into scaffold sequence.

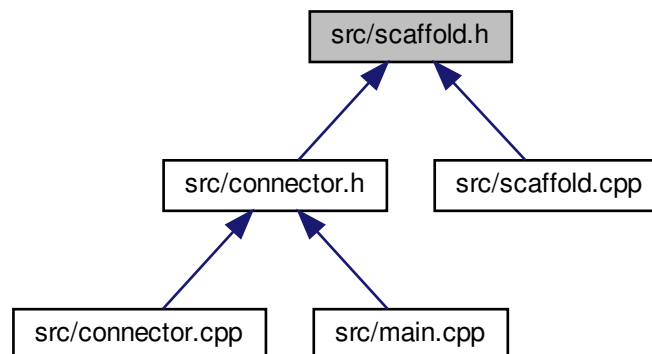
8.17 src/scaffold.h File Reference

Header file for [Scaffold](#) class.

```
#include <vector>
#include <unordered_set>
#include <utility>
#include <string>
#include "../contig.h"
Include dependency graph for scaffold.h:
```



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



Classes

- class [Scaffold](#)

[Scaffold](#) class is representation of scaffold structure.

8.17.1 Detailed Description

Header file for [Scaffold](#) class.

Author

Marko Culinovic marko.culinovic@gmail.com

Luka Sterbic luka.sterbic@gmail.com

Header file for [Scaffold](#) class. It is used as representation of scaffolds. [Scaffold](#) consists of multiple Contigs and memorizes contributions of each contig into scaffold sequence.

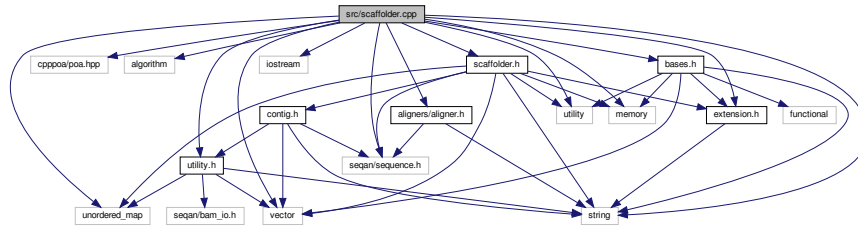
8.18 src/scaffolder.cpp File Reference

Implementation file for scaffolder namespace.

```
#include <seqan/sequence.h>
#include <cpppoa/poa.hpp>
#include <algorithm>
#include <vector>
#include <string>
#include <iostream>
#include <utility>
#include <memory>
#include <unordered_map>
```

```
#include "aligners/aligner.h"
#include "utility.h"
#include "scaffolder.h"
#include "extension.h"
#include "bases.h"
```

Include dependency graph for scaffolder.cpp:



Namespaces

- [scaffolder](#)

Scaffolder namespace provides functions for different methods of contig extension.

Macros

- `#define INNER_MARGIN 5`
- `#define OUTER_MARGIN 15`
- `#define MIN_COVERAGE 5`

Functions

- void [scaffolder::set_max_extension_len](#) (int length)
Methods sets maximum extension length.
- void [scaffolder::set_inner_margin](#) (int margin)
Sets the inner margin.
- void [scaffolder::set_outer_margin](#) (int margin)
Sets the outhter margin.
- void [scaffolder::set_min_coverage](#) (int coverage)
Method sets the minimum coverage.
- void [scaffolder::find_possible_extensions](#) (const vector< BamAlignmentRecord > &aln_records, vector< shared_ptr< Extension >> *pleft_ext_reads, vector< shared_ptr< Extension >> *pright_ext_reads, const unordered_map< string, uint32_t > &read_name_to_id, uint64_t contig_len)
- string [scaffolder::get_extension_mv_simple](#) (const vector< shared_ptr< Extension >> &extensions)
- string [scaffolder::get_extension_mv_realign](#) (const vector< shared_ptr< Extension >> &extensions)
Method finds contig extension using majority vote on each position of possible extensions while coverage >= k and tries to realign reads which base isn't elected as majority.
- Contig * [scaffolder::extend_contig](#) (Dna5String &contig_seq, const vector< BamAlignmentRecord > &aln_records, const unordered_map< string, uint32_t > &read_name_to_id, const StringSet< CharString > &read_ids, const StringSet< Dna5String > &read_seqs)
Method tries to extend contig using global realignment method on both sides with given alignment records.
- Contig * [scaffolder::extend_contig_poa](#) (const Dna5String &contig_seq, const vector< BamAlignmentRecord > &aln_records, const unordered_map< string, uint32_t > &read_name_to_id)
Method tries to extend contig using POA consensus method on both sides with given alignment records.

Variables

- int `scaffolder::max_ext_length` = 1000
- int `scaffolder::inner_margin` = 5
- int `scaffolder::outer_margin` = 15
- int `scaffolder::min_coverage` = 5
- const char * `scaffolder::tmp_contig_file` = "tmp/extend_contig.fasta"
- const char * `scaffolder::tmp_reads_file` = "tmp/realign_reads.fasta"
- const char * `scaffolder::tmp_sam_file` = "tmp/realign.sam"

8.18.1 Detailed Description

Implementation file for scaffolder namespace.

Copyright

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

Implementation file for scaffolder namespace. It provides functions for different methods of contig extension.

8.18.2 Macro Definition Documentation

8.18.2.1 INNER_MARGIN

```
#define INNER_MARGIN 5
```

8.18.2.2 MIN_COVERAGE

```
#define MIN_COVERAGE 5
```

8.18.2.3 OUTER_MARGIN

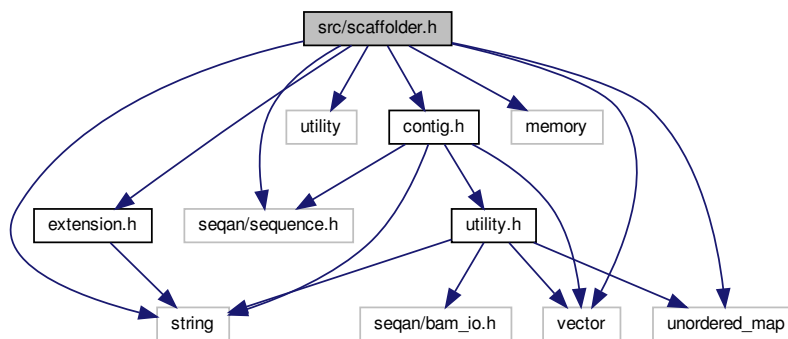
```
#define OUTER_MARGIN 15
```

8.19 src/scaffolder.h File Reference

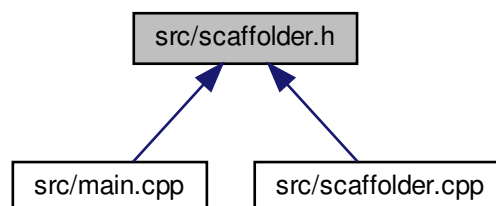
Header file for scaffolder namespace.

```
#include <seqan/sequence.h>
#include <vector>
#include <string>
#include <utility>
#include <unordered_map>
#include <memory>
#include "extension.h"
#include "contig.h"
```

Include dependency graph for scaffolder.h:



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



Namespaces

- [scaffolder](#)

Scaffolder namespace provides functions for different methods of contig extension.

Functions

- void `scaffolder::set_max_extension_len` (int length)
Methods sets maximum extension length.
- void `scaffolder::set_inner_margin` (int margin)
Sets the inner margin.
- void `scaffolder::set_outer_margin` (int margin)
Sets the outer margin.
- void `scaffolder::set_min_coverage` (int coverage)
Method sets the minimum coverage.
- void `scaffolder::find_possible_extensions` (const vector< BamAlignmentRecord > &aln_records, vector< string > *pleft_extensions, vector< string > *pright_extensions, uint64_t contig_len)
Method finds substrings of reads which extend contig on both ends.
- string `scaffolder::get_extension_mv_simple` (const vector< string > &extensions)
Method finds contig extension using majority vote on each position of possible extensions while coverage $\geq k$.
- string `scaffolder::get_extension_mv_realign` (const vector< shared_ptr< Extension > > &extensions)
Method finds contig extension using majority vote on each position of possible extensions while coverage $\geq k$ and tries to realign reads which base isn't elected as majority.
- Contig * `scaffolder::extend_contig` (Dna5String &contig_seq, const vector< BamAlignmentRecord > &aln_records, const unordered_map< string, uint32_t > &read_name_to_id, const StringSet< CharString > &read_ids, const StringSet< Dna5String > &read_seqs)
Method tries to extend contig using global realignment method on both sides with given alignment records.
- Contig * `scaffolder::extend_contig_poa` (const Dna5String &contig_seq, const vector< BamAlignmentRecord > &aln_records, const unordered_map< string, uint32_t > &read_name_to_id)
Method tries to extend contig using POA consensus method on both sides with given alignment records.

8.19.1 Detailed Description

Header file for scaffolder namespace.

Copyright

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

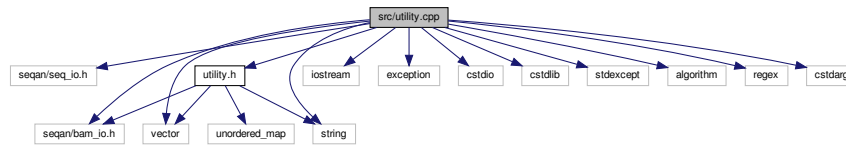
Header file for scaffolder namespace. It provides functions for different methods of contig extension.

8.20 src/utility.cpp File Reference

Implementation of various utility functions.

```
#include <seqan/seq_io.h>
#include <seqan/bam_io.h>
#include <iostream>
#include <exception>
#include <cstdio>
#include <cstdlib>
#include <vector>
#include <string>
#include <stdexcept>
```

```
#include <algorithm>
#include <regex>
#include <cstdlib>
#include "utility.h"
Include dependency graph for utility.cpp:
```



Namespaces

- [utility](#)

Functions

- unsigned int [utility::get_concurrency_level](#) ()
Gets the concurrency level.
- void [utility::set_concurrency_level](#) (int threads)
Sets the concurrency level.
- void [utility::read_fasta](#) (StringSet< CharString > *pids, StringSet< Dna5String > *pseqs, char *ont_reads, char *_filename)
Reads a FASTA file.
- void [utility::write_fasta](#) (const CharString &id, const Dna5String &seq, const char *filename)
Write a sequence to file.
- void [utility::write_fasta](#) (const StringSet< CharString > &ids, const StringSet< Dna5String > &seqs, const char *filename)
Writes a set of sequences to file.
- void [utility::read_sam](#) (BamHeader *pheader, vector< BamAlignmentRecord > *precords, const char *filename)
Read a SAM file.
- void [utility::read_sam](#) (BamFileIn *pinput_file, BamHeader *pheader, vector< BamAlignmentRecord > *precords)
Read a SAM file.
- void [utility::map_alignments](#) (const char *filename, AlignmentCollection *collection, const unordered_map< string, uint32_t > &contig_name_to_id)
Reads and map aligned reads to contigs.
- void [utility::execute_command](#) (const char *format,...)
Run shell command.
- int [utility::base_to_idx](#) (char base)
Char base to int id.
- char [utility::idx_to_base](#) (int idx)
Base id to char base.
- void [utility::exit_with_message](#) (const char *format,...)
Prints error message and exits.
- string [utility::CharString_to_string](#) (const CharString &str)
Converts seqan::CharString to std::string.
- string [utility::Dna5String_to_string](#) (const Dna5String &str)

- Converts seqan::Dna5String to std::string.*

 - int `utility::contributes_to_seq_len` (char c)
Method used to determine read length from cigar string.
 - int `utility::contributes_to_contig_len` (char c)
Method used to determine length of contig part to which read is aligned.
 - string `utility::reverse_complement` (const Dna5String &seq)
Method creates reverse complement from string given as parameter.
 - string `utility::create_seq_id` (const char *format,...)
Helper method for creating sequence ID.
 - bool `utility::is_command_available` (const char *command)
Checks if the given command is available through the system shell.

Variables

- unsigned int `utility::hardware_concurrency`
- char `utility::command_buffer` [COMMAND_BUFFER_SIZE] = { 0 }
Buffer to hold shell command strings.
- char `utility::error_buffer` [ERROR_BUFFER_SIZE] = { 0 }
Buffer to hold a description string when an error occurs.
- char `utility::seq_id_buffer` [SEQ_ID_BUFFER_SIZE] = { 0 }
Buffer used to build the name of a sequence.

8.20.1 Detailed Description

Implementation of various utility functions.

Copyright

Marko Culinovic marko.culinovic@fer.hr
 Luka Sterbic luka.sterbic@fer.hr

Implementation of utility functions for genomic data I/O, shell commands execution and data conversion.

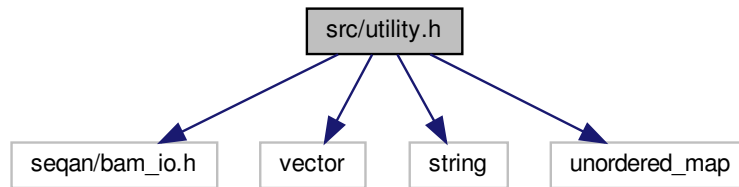
8.21 src/utility.h File Reference

Various utility functions.

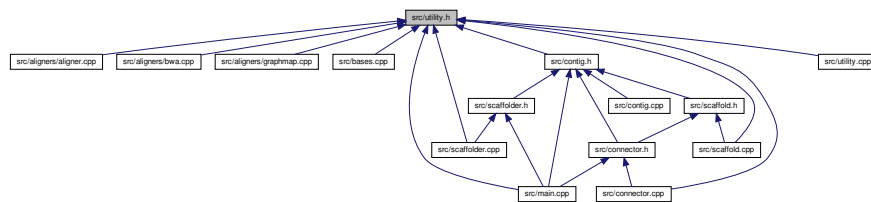
```
#include <seqan/bam_io.h>
#include <vector>
#include <string>
```

```
#include <unordered_map>
```

Include dependency graph for utility.h:



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



Namespaces

- [utility](#)

Macros

- `#define` [DEBUG\(x\)](#)
- `#define` [DEBUG_VAR\(x\)](#)
- `#define` [DEBUG_BLOCK\(x\)](#)
- `#define` [MINIMUM_CONTIG_LEN](#) 30000
- `#define` [UNMAPPED](#) 0x4
SAM format unmapped read flag.
- `#define` [COMMAND_BUFFER_SIZE](#) 256
The size of the shell command buffer in bytes.
- `#define` [ERROR_BUFFER_SIZE](#) 160
The size of the shell command buffer in bytes.
- `#define` [SEQ_ID_BUFFER_SIZE](#) 160
The size of the sequence name buffer in bytes.

Typedefs

- `typedef unordered_map< int, vector< BamAlignmentRecord > >` [AlignmentCollection](#)
Structure used to cluster read alignments to specific contigs.

Functions

- unsigned int [utility::get_concurrency_level](#) ()
Gets the concurrency level.
- void [utility::set_concurrency_level](#) (int threads)
Sets the concurrency level.
- void [utility::read_fasta](#) (StringSet< CharString > *pids, StringSet< Dna5String > *pseqs, char *ont_reads←_filename)
Reads a FASTA file.
- void [utility::write_fasta](#) (const CharString &id, const Dna5String &seq, const char *filename)
Write a sequence to file.
- void [utility::write_fasta](#) (const StringSet< CharString > &ids, const StringSet< Dna5String > &seqs, const char *filename)
Writes a set of sequences to file.
- void [utility::read_sam](#) (BamHeader *pheader, vector< BamAlignmentRecord > *precords, const char *filename)
Read a SAM file.
- void [utility::map_alignments](#) (const char *filename, [AlignmentCollection](#) *collection, const unordered_map< string, uint32_t > &contig_name_to_id)
Reads and map aligned reads to contigs.
- void [utility::execute_command](#) (const char *format,...)
Run shell command.
- int [utility::base_to_idx](#) (char base)
Char base to int id.
- char [utility::idx_to_base](#) (int idx)
Base id to char base.
- template<typename T >
void [utility::throw_exception](#) (const char *format,...)
Throw an exception.
- void [utility::exit_with_message](#) (const char *format,...)
Prints error message and exits.
- string [utility::CharString_to_string](#) (const CharString &str)
Converts seqan::CharString to std::string.
- string [utility::Dna5String_to_string](#) (const Dna5String &str)
Converts seqan::Dna5String to std::string.
- int [utility::contributes_to_seq_len](#) (char c)
Method used to determine read length from cigar string.
- int [utility::contributes_to_contig_len](#) (char c)
Method used to determine length of contig part to which read is aligned.
- string [utility::reverse_complement](#) (const Dna5String &seq)
Method creates reverse complement from string given as parameter.
- string [utility::create_seq_id](#) (const char *format,...)
Helper method for creating sequence ID.
- bool [utility::is_command_available](#) (const char *command)
Checks if the given command is available through the system shell.

8.21.1 Detailed Description

Various utility functions.

Copyright

Marko Culinovic marko.culinovic@gmail.com
Luka Sterbic luka.sterbic@gmail.com

Header file with declaration of utility functions for genomic data I/O, shell commands execution and data conversion.

8.21.2 Macro Definition Documentation

8.21.2.1 COMMAND_BUFFER_SIZE

```
#define COMMAND_BUFFER_SIZE 256
```

The size of the shell command buffer in bytes.

8.21.2.2 DEBUG

```
#define DEBUG(  
    x )
```

8.21.2.3 DEBUG_BLOCK

```
#define DEBUG_BLOCK(  
    x )
```

8.21.2.4 DEBUG_VAR

```
#define DEBUG_VAR(  
    x )
```

8.21.2.5 ERROR_BUFFER_SIZE

```
#define ERROR_BUFFER_SIZE 160
```

The size of the shell command buffer in bytes.

8.21.2.6 MINIMUM_CONTIG_LEN

```
#define MINIMUM_CONTIG_LEN 30000
```


8.21.2.7 SEQ_ID_BUFFER_SIZE

```
#define SEQ_ID_BUFFER_SIZE 160
```

The size of the sequence name buffer in bytes.

8.21.2.8 UNMAPPED

```
#define UNMAPPED 0x4
```

SAM format unmapped read flag.

8.21.3 Typedef Documentation

8.21.3.1 AlignmentCollection

```
typedef unordered_map<int, vector<BamAlignmentRecord> > AlignmentCollection
```

Structure used to cluster read alignments to specific contigs.

Index

- ~Aligner
 - Aligner, [33](#)
- ~Connector
 - Connector, [42](#)
- ~GraphMapAligner
 - GraphMapAligner, [53](#)
- ANCHOR_LEN
 - contig.h, [75](#)
- ANCHOR_THRESHOLD
 - connector.h, [72](#)
- add_contig
 - Scaffold, [56](#)
- align
 - Aligner, [33](#), [34](#)
 - BwaAligner, [39](#), [40](#)
 - GraphMapAligner, [54](#)
- Aligner, [31](#)
 - ~Aligner, [33](#)
 - align, [33](#), [34](#)
 - Aligner, [32](#)
 - get_instance, [34](#)
 - get_name, [34](#)
 - get_tmp_alignment_filename, [34](#)
 - get_tmp_contig_filename, [34](#)
 - get_tmp_reference_filename, [34](#)
 - index, [34](#)
 - init, [35](#)
 - name, [35](#)
 - tech_type, [35](#)
- AlignmentCollection
 - utility.h, [93](#)
- base_to_idx
 - utility, [21](#)
- bases, [11](#)
 - count_bases, [11](#), [12](#)
- bases.h
 - bool_predicate, [69](#)
 - NUM_BASES, [69](#)
- bases::BasesCounter, [35](#)
 - BasesCounter, [36](#)
 - count, [37](#)
 - coverage, [37](#)
 - digest_base, [36](#)
 - max_idx, [37](#)
 - refresh_stats, [37](#)
- BasesCounter
 - bases::BasesCounter, [36](#)
- bool_predicate
 - bases.h, [69](#)
- BwaAligner, [38](#)
 - align, [39](#), [40](#)
 - BwaAligner, [39](#)
 - index, [41](#)
- COMMAND_BUFFER_SIZE
 - utility.h, [92](#)
- COMPLEMENT
 - connector.h, [72](#)
- CharString_to_string
 - utility, [21](#)
- command_buffer
 - utility, [28](#)
- connect_contigs
 - Connector, [42](#)
- Connector, [41](#)
 - ~Connector, [42](#)
 - connect_contigs, [42](#)
 - Connector, [42](#)
 - dump_scaffolds, [43](#)
 - get_scaffolds, [43](#)
- connector.h
 - ANCHOR_THRESHOLD, [72](#)
 - COMPLEMENT, [72](#)
 - SECONDARY_LINE, [72](#)
 - UNMAPPED, [72](#)
- contains
 - Scaffold, [56](#)
- Contig, [43](#)
 - Contig, [45](#)
 - dump_anchors, [46](#)
 - ext_left, [46](#)
 - ext_right, [46](#)
 - id, [46](#)
 - left_id, [47](#)
 - operator!=, [47](#)
 - operator==, [47](#)
 - reverse_complement, [47](#)
 - right_ext_pos, [48](#)
 - right_id, [48](#)
 - seq, [48](#)
 - set_id, [48](#)
 - total_ext_left, [49](#)
 - total_ext_right, [49](#)
 - total_len, [49](#)
- contig.h
 - ANCHOR_LEN, [75](#)
- contigs_filename
 - main.cpp, [80](#)

- contributes_to_contig_len
 - utility, [22](#)
- contributes_to_seq_len
 - utility, [22](#)
- count
 - bases::BasesCounter, [37](#)
- count_bases
 - bases, [11](#), [12](#)
- coverage
 - bases::BasesCounter, [37](#)
- create_seq_id
 - utility, [22](#)
- curr_pos
 - Extension, [51](#)
- DEBUG_BLOCK
 - utility.h, [92](#)
- DEBUG_VAR
 - utility.h, [92](#)
- DEBUG
 - utility.h, [92](#)
- digest_base
 - bases::BasesCounter, [36](#)
- Dna5String_to_string
 - utility, [23](#)
- do_operation
 - Extension, [51](#)
- draft_genome_filename
 - main.cpp, [80](#)
- dump_anchors
 - Contig, [46](#)
- dump_scaffolds
 - Connector, [43](#)
- ERROR_BUFFER_SIZE
 - utility.h, [92](#)
- error_buffer
 - utility, [29](#)
- execute_command
 - utility, [23](#)
- exit_with_message
 - utility, [24](#)
- ext_left
 - Contig, [46](#)
- ext_right
 - Contig, [46](#)
- extend_contig
 - scaffolder, [14](#)
- extend_contig_poa
 - scaffolder, [15](#)
- Extension, [50](#)
 - curr_pos, [51](#)
 - do_operation, [51](#)
 - Extension, [50](#)
 - is_dropped, [52](#)
 - read_id, [51](#)
 - seq, [51](#)
- extension.h
 - Operation, [77](#)
- extensions_filename
 - main.cpp, [80](#)
- find_possible_extensions
 - scaffolder, [15](#)
- first_contig
 - Scaffold, [57](#)
- get_combined_sequence
 - Scaffold, [57](#)
- get_concurrency_level
 - utility, [24](#)
- get_contigs
 - Scaffold, [57](#)
- get_extension_mv_realign
 - scaffolder, [16](#)
- get_extension_mv_simple
 - scaffolder, [16](#), [17](#)
- get_instance
 - Aligner, [34](#)
- get_name
 - Aligner, [34](#)
- get_scaffolds
 - Connector, [43](#)
- get_tmp_alignment_filename
 - Aligner, [34](#)
- get_tmp_contig_filename
 - Aligner, [34](#)
- get_tmp_reference_filename
 - Aligner, [34](#)
- GraphMapAligner, [52](#)
 - ~GraphMapAligner, [53](#)
 - align, [54](#)
 - GraphMapAligner, [53](#)
 - index, [54](#)
- hardware_concurrency
 - utility, [29](#)
- INNER_MARGIN
 - scaffolder.cpp, [85](#)
- id
 - Contig, [46](#)
- idx_to_base
 - utility, [24](#)
- index
 - Aligner, [34](#)
 - BwaAligner, [41](#)
 - GraphMapAligner, [54](#)
- init
 - Aligner, [35](#)
- inner_margin
 - scaffolder, [18](#)
- is_command_available
 - utility, [25](#)
- is_dropped
 - Extension, [52](#)
- last_contig

- Scaffold, 58
- left_id
 - Contig, 47
- MIN_COVERAGE
 - scaffolder.cpp, 85
- MINIMUM_CONTIG_LEN
 - utility.h, 92
- main
 - main.cpp, 79
- main.cpp
 - contigs_filename, 80
 - draft_genome_filename, 80
 - extensions_filename, 80
 - main, 79
 - PATH_BUFFER_SIZE, 79
 - RELEASE_DATE, 79
 - reads_filename, 80
 - scaffolds_filename, 80
 - set_output_paths, 79
 - setup_cmd_interface, 79
 - tmp_dirname, 80
 - trim_circular_genome, 80
 - use_POA_consensus, 81
 - use_graphmap_aligner, 81
 - use_tech_type, 81
 - VERSION, 79
- map_alignments
 - utility, 25
- max_ext_length
 - scaffolder, 18
- max_idx
 - bases::BasesCounter, 37
- merge
 - Scaffold, 58
- min_coverage
 - scaffolder, 19
- NUM_BASES
 - bases.h, 69
- name
 - Aligner, 35
- num_contigs
 - Scaffold, 58
- OUTER_MARGIN
 - scaffolder.cpp, 85
- Operation
 - extension.h, 77
- operator!=
 - Contig, 47
- operator==
 - Contig, 47
- outer_margin
 - scaffolder, 19
- PATH_BUFFER_SIZE
 - main.cpp, 79
- RELEASE_DATE
 - main.cpp, 79
- read_fasta
 - utility, 26
- read_id
 - Extension, 51
- read_sam
 - utility, 26
- read_type, 12
 - ReadType, 12
 - string_to_read_type, 13
- ReadType
 - read_type, 12
- reads_filename
 - main.cpp, 80
- refresh_stats
 - bases::BasesCounter, 37
- reverse_complement
 - Contig, 47
 - utility, 26
- right_ext_pos
 - Contig, 48
- right_id
 - Contig, 48
- SECONDARY_LINE
 - connector.h, 72
- SEQ_ID_BUFFER_SIZE
 - utility.h, 92
- Scaffold, 55
 - add_contig, 56
 - contains, 56
 - first_contig, 57
 - get_combined_sequence, 57
 - get_contigs, 57
 - last_contig, 58
 - merge, 58
 - num_contigs, 58
 - Scaffold, 56
 - trim, 58
- scaffolder, 13
 - extend_contig, 14
 - extend_contig_poa, 15
 - find_possible_extensions, 15
 - get_extension_mv_realign, 16
 - get_extension_mv_simple, 16, 17
 - inner_margin, 18
 - max_ext_length, 18
 - min_coverage, 19
 - outer_margin, 19
 - set_inner_margin, 17
 - set_max_extension_len, 17
 - set_min_coverage, 18
 - set_outer_margin, 18
 - tmp_contig_file, 19
 - tmp_reads_file, 19
 - tmp_sam_file, 19
- scaffolder.cpp
 - INNER_MARGIN, 85
 - MIN_COVERAGE, 85

- OUTER_MARGIN, 85
- scaffolds_filename
 - main.cpp, 80
- seq
 - Contig, 48
 - Extension, 51
- seq_id_buffer
 - utility, 29
- set_concurrency_level
 - utility, 27
- set_id
 - Contig, 48
- set_inner_margin
 - scaffolder, 17
- set_max_extension_len
 - scaffolder, 17
- set_min_coverage
 - scaffolder, 18
- set_outer_margin
 - scaffolder, 18
- set_output_paths
 - main.cpp, 79
- setup_cmd_interface
 - main.cpp, 79
- src/aligners/aligner.cpp, 61
- src/aligners/aligner.h, 62
- src/aligners/bwa.cpp, 63
- src/aligners/bwa.h, 64
- src/aligners/graphmap.cpp, 65
- src/aligners/graphmap.h, 65
- src/bases.cpp, 67
- src/bases.h, 68
- src/connector.cpp, 70
- src/connector.h, 71
- src/contig.cpp, 73
- src/contig.h, 74
- src/extension.cpp, 75
- src/extension.h, 76
- src/main.cpp, 77
- src/scaffold.cpp, 81
- src/scaffold.h, 82
- src/scaffolder.cpp, 83
- src/scaffolder.h, 86
- src/utility.cpp, 87
- src/utility.h, 89
- string_to_read_type
 - read_type, 13
- tech_type
 - Aligner, 35
- throw_exception
 - utility, 27
- tmp_contig_file
 - scaffolder, 19
- tmp_dirname
 - main.cpp, 80
- tmp_reads_file
 - scaffolder, 19
- tmp_sam_file
 - scaffolder, 19
- total_ext_left
 - Contig, 49
- total_ext_right
 - Contig, 49
- total_len
 - Contig, 49
- trim
 - Scaffold, 58
- trim_circular_genome
 - main.cpp, 80
- UNMAPPED
 - connector.h, 72
 - utility.h, 93
- use_POA_consensus
 - main.cpp, 81
- use_graphmap_aligner
 - main.cpp, 81
- use_tech_type
 - main.cpp, 81
- utility, 20
 - base_to_idx, 21
 - CharString_to_string, 21
 - command_buffer, 28
 - contributes_to_contig_len, 22
 - contributes_to_seq_len, 22
 - create_seq_id, 22
 - Dna5String_to_string, 23
 - error_buffer, 29
 - execute_command, 23
 - exit_with_message, 24
 - get_concurrency_level, 24
 - hardware_concurrency, 29
 - idx_to_base, 24
 - is_command_available, 25
 - map_alignments, 25
 - read_fasta, 26
 - read_sam, 26
 - reverse_complement, 26
 - seq_id_buffer, 29
 - set_concurrency_level, 27
 - throw_exception, 27
 - write_fasta, 28
- utility.h
 - AlignmentCollection, 93
 - COMMAND_BUFFER_SIZE, 92
 - DEBUG_BLOCK, 92
 - DEBUG_VAR, 92
 - DEBUG, 92
 - ERROR_BUFFER_SIZE, 92
 - MINIMUM_CONTIG_LEN, 92
 - SEQ_ID_BUFFER_SIZE, 92
 - UNMAPPED, 93
- VERSION
 - main.cpp, 79
- write_fasta

utility, [28](#)