Analysis Software Package

Generated by Doxygen 1.8.11

Contents

1 Format of input files for all utilities **2**

1.1 Structure file 2

1.2 Optional bond file 3

1.3 Ordered coordinate file 4

1.4 Indexed coordinate file 4

1.5 Aggregate file 4

2 Options for all utilities **5**

3 Common utilities **5**

3.1 traject utility 5

3.2 SelectedVcf utility 6

3.3 Config utility 6

3.4 TransformVsf utility 7

3.4.1 Format of output structure file 7

3.5 Aggregates utility 8

4 Utilities for linear chains **9**

4.1 EndToEnd utility 9

4.2 PersistenceLength utility 9

5 Utilities for nanoparticles **10**

5.1 C60 utility 10

6 Todo List **10**

7 Data Structure Index **11**

7.1 Data Structures 11

8 File Index **11**

8.1 File List 11

9 Data Structure Documentation **11**

9.1 Aggregate Struct Reference 11

9.2 Bead Struct Reference 12

9.3 BeadType Struct Reference 12

9.4 Counts Struct Reference 13

9.5 Molecule Struct Reference 13

9.6 MoleculeType Struct Reference 13

9.7 Vector Struct Reference 14

10 File Documentation **14**

10.1 AnalysisTools.h File Reference 14

10.1.1 Function Documentation 15

1 Format of input files for all utilities

All the utilities read information about studied system from `vsf/vcf files` (formatted as described below) and `FIELD` file (input file for `DL_MESO simulation package`). Coordinates are read from a `.vcf` file (with either ordered timesteps or indexed timesteps). Structure of the system (names and numbers of beads and molecules, etc.) is read from `FIELD` file and `.vsf` files, but only bead types that are in the above mentioned `.vcf` file are considered.

Aggregate file is of my own format and is used by every utility doing calculation on whole aggregates (as opposed to calculations on individual molecules).

1.1 Structure file

The software package is designed with file `dl_meso.vsf` in mind, which is generated by the `traject` utility provided in DL_MESO software (and modified by me). Generally, the utilities are tested only against files generated by `traject`, but other `.vsf` files (such as the one generated by `TransformVsf` utility) should work fine, if formatted according to the following guidelines.

The first mandatory line specifies default bead type which means all atom lines for beads of this type are unnecessary (provided those beads are not in a molecule). All atom lines in `dl_meso.vsf` specify VDW radius and atom name. If an atom is in a molecule, its molecule number is appended to the atom line as `resid <id>`:

```
atom default radius 1.000000 name <name>
...
atom <id> radius 1.000000 name <name> resid <id>
...
```

Only the bead number and name are read, so both VDW radius and molecule number are not strictly necessary. Short version of `atom` and `name` keywords (`a` and `n` respectively) can be used. Other keywords can be included, because they will be ignored. No comments are allowed in `.vsf` file.

Bond lines of `.vsf` files are not read and are therefore irrelevant to all the utilities.

1.2 Optional bond file

Bonds for each molecule type are specified in `FIELD`, but they can be read from a different file if required.

The file with molecule bonds must contain name of the molecule type (same as in `FIELD`) followed by number of bonds on the next line and on every subsequent line two index numbers corresponding to the two connected beads (bead numbers start from one and are ordered according to beads in `FIELD` section for the given molecule). The bead numbers do not have to be sorted in any way and no blank lines should be present.

Example of bond file:

```
triangle
3
1 2 possible
3 1
2 3 comment
```

This file must be used for molecule types that have only some of its beads in `.vcf` file with indexed timesteps. In such a case, the bead indices correspond to `FIELD` as if the bead types not present in `.vcf` are not present `FIELD`.

Example of the relevant part of `FIELD`:

```
...
beads 3
A <float> <float> <float>
B <float> <float> <float>
A <float> <float> <float>
bonds 3
harm 1 2 <float> <float>
harm 1 3 <float> <float>
harm 2 3 <float> <float>
finish
```

Assuming only bead types `A` are present in `.vcf` file, the now necessary bond file would like like this:

```
name
1
1 2 possible comment
```

Should the bond file not be provided such case, the utilities detect no error, but will not work correctly (and may crash with segmentation fault).

Bond information about molecule types not present in the bond file will be read from `FIELD`.

1.3 Ordered coordinate file

First line of `.vcf` file with ordered timestep(s) contains box size. Each timestep starts with a comment line (i.e. line starting with `#` sign), the second line contains `timestep` (or the short version, `t`) and each following line contains the coordinates of a single bead. Every bead from `.vsf` structure file must be present in each timestep.

Exactly one blank line must be between every two timesteps and no blank lines are allowed at the end of the file.

Example of ordered coordinate file:

```
pbc <float> <float> <float>
<blank line>
# 1
timestep
<float> <float> <float>
...
```

1.4 Indexed coordinate file

Unlike the `.vcf` file with ordered timesteps, the `.vcf` file with indexed timestep does not contain coordinates for every bead. Only beads of selected bead types are present and their names are written as comments at the beginning of the file (and followed by a blank line). Every bead is prepended by its index number according to the `.vsf` structure file. Keyword `timestep` (or `t`) at the beginning of every timestep is replaced by `indexed` (or its short version, `i`). Otherwise the file has the same format as the `.vcf` file with ordered timesteps.

Example of indexed coordinate file:

```
# name
...
<blank line>
pbc <float> <float> <float>
<blank line>
# 1
indexed
<id> <float> <float> <float>
...
```

1.5 Aggregate file

The aggregate file with `.agg` ending is generated using Aggregates utility. It contains information about the number of aggregates in the system in every simulation timestep and therefore is linked to the `.vcf` file used to calculate the aggregates. For every aggregate in each timestep there is a number and ids of molecules in that aggregate as well as a number and ids of monomeric beads near the aggregate.

The first line of an aggregate file contains the command used to generate it. The subsequent lines contain information on individual timesteps starting with `Step` keyword, followed by the number of aggregates in the timestep and followed by individual aggregates. Every aggregate is spread over two lines - the first one contains the number of molecules in the aggregate followed by their ids (according to a corresponding `.vsf` structure file) and the second line contains the number of monomeric beads in the aggregate followed by its ids (again, the ids correspond to the `.vsf` file). The line with monomeric beads is indented for easier reading.

Example of an aggregate file:

```
<command used to generate it>
<blank line>
Step:  1
<number of aggregates in step 1>
<blank line>
2 :  1 34
3 :  230 40000 41003
<number of molecules in the second aggregate> :  <molecule ids>
<number of monomeric beads in the aggregate :  <bead ids>
<blank line>
Step:  2
...
<blank line>
Last Step:  <number>
```

2 Options for all utilities

A great majority of utilities has several options that are the same. These options are described here, but can be used with any utility unless stated otherwise.

`-i <name>`
use custom `.vsf` structure file instead of the default `dl_meso.vsf` (must end with `.vsf`)

`-b <name>`
file containing bond alternatives to `FIELD` (see Optional bond file for explanation)

`-v`
verbose output providing information about the system

`-V`
more detailed verbose output (also prints comments from `.vcf` file at the start of every timestep)

`-h`
print help and exit

3 Common utilities

Utilies that are not specific to any given system and are used for all simulations.

3.1 traject utility

This utility is from the `DL_MESO simulation package`. While originally it creates a `.vtf` file containing both structure and coordinates, I have changed it to create a separate `dl_meso.vsf` structure file and `All.vcf` coordinate file containing ordered timesteps.

Usage:

```
traject <cores>
```

`<cores>`
number of computer cores used for the simulation run (or the number of `HISTORY` file)

The standard options cannot be used with this utility.

3.2 SelectedVcf utility

This utility takes `.vcf` file containing either ordered timesteps (such as `All.vcf` created by DL_MESO `traject` utility which was modified by me) or indexed timesteps and creates a new `.vcf` coordinate file containing only beads of selected types with an option of removing periodic boundary condition and thus joining molecules. The otput `.vcf` file therefore contains indexed timesteps.

Usage:

```
SelectedVcf <input.vcf> <start> <skip> <output.vcf> <type names> <options>
```

`<input.vcf>`
: input coordinate filename (must end with `.vcf`) containing either ordered or indexed timesteps

`<start>`
: number of timestep to start from

`<skip>`
: leave out every `skip` steps

`<output.vcf>`
: output filename with indexed coordinates (must end with `.vcf`)

`<type names>`
: names of bead types to save

`<options>`
: `-j`
  : join individual molecules by removing periodic boundary conditions

Todo Implement `<skip>` and `<start>`

3.3 Config utility

This utility takes `.vcf` file containing either ordered timesteps (such as `All.vcf` created by DL_MESO `traject` utility which was modified by me) or indexed timesteps and creates `CONFIG` file (file containing initial coordinates for a simulation via `DL_MESO simulation package`).

Usage:

```
Config <input.vcf> <options>
```

`<input.vcf>`
: input coordinate filename (must end with `.vcf`) containing either ordered or indexed timesteps

Todo Implement possibility to choose timestep number for creating `CONFIG` file.

3.4 TransformVsf utility

This utility takes `.vsf` structure file and DL_MESO input file `FIELD` and transforms them into a different `.vsf` structure file that is well suited for visualisation using VMD software.

Usage:

```
TransformVsf <output.vsf> <options>
```

`<output.vsf>`

output structure file that must end with `.vsf`

3.4.1 Format of output structure file

Every atom line in the generated structure file contains bead's index number, mass, charge and name. Atom lines for beads in molecules also contain molecule's id number and the name of the type of molecule. The bond section of `output.vsf` lists all bonds one by one (i.e. no chains of bonds in the format `<id1>:: <id2>` are used). Information about which bonds belong to which molecule is provided as has comment. The file has the following format:

```
atom default name <name> mass <m> charge <q>
...
atom <id> name <name> mass <m> charge <q>
...
atom <id> name <name> mass <m> charge <q> segid <name> resid
<id>
...
# resid <id>
<bonded bead id1>: <bonded bead id2>
...
```

For VMD atom selection:

`segid <name>`

selects all molecules with given name(s)

`resid <id>`

selects molecule(s) with given index number(s)

`charge <q>`

selects all beads with given charge(s) (double quotes are required for negative charge)

`mass <m>`

selects all beads with given mass(es)

Todo Somehow avoid the need to use the special optional bond file, where the ids of beads must strictly adhere to FIELD. Possibly require use of a vcf file in conjunction with bond file

3.5 Aggregates utility

This utility determines which molecules belong to which aggregates according to a simple criterion: two molecules belong to the same aggregate if they at least a specified number of contact pairs. A contact pair is a pair of two beads belonging to different molecules which are closer than certain distance. Both the distance and the number of needed contact pairs are arguments of the command.

Usage:

```
Aggregates <input.vcf> <distance> <contacts> <output.agg> <type names>
<options>
```

`<input.vcf>`

input coordinate filename (must end with `.vcf`) containing either ordered or indexed timesteps

`<output.agg>`

output filename (must end with `.agg`) containing information about aggregates

`<distance>`

minimum distance for two beads to be in contact (constituting one contact pair)

`<contacts>`

minimum number of contact pairs to consider two molecules to be in one aggregate

`<type names>`

names of bead types to use for calculating contact pairs

`<options>`

`-j <joined.vcf>`
filename for coordinates of joined aggregates (must end with `.vcf`)

Todo Add the possibility to save only certain bead types to output vcf file with joined coordinates.

Average

Utility calculating average values with standard deviation and autocorrelation time from values contained in a text file. The first line of the file has to contain the number of data lines and no comments are allowed.

Usage:

```
Average <filename> <column> <discard> <n_blocks>
```

`<filename>`

name of data filel

`<column>`

column number in the file containing the data to analyze

`<discard>`

number of data values considered as equilibrium

`<n_blocks>`

number of blocks for binning analysis

4 Utilities for linear chains

This section provides information about utilities with calculations that are sensible to do only on linear polymer chains

4.1 EndToEnd utility

This utility calculates end-to-end distance of specified molecules. End-to-end distance makes sense only for linear chains, therefore it is assumed that the provided molecule names are linear chains. No check is performed. The distance is calculated between the first and the last bead of the molecule; that is, between the first and the last bead in the `FIELD` entry for the given molecule. Also the use of joined coordinates (that is, without periodic boundary condition) is required, because the utility does not remove periodic boundary conditions.

The output is a file containing average end-to-end distance for every molecule type for each timestep.

Usage:

```
EndToEnd <input.vcf> <output.vcf> <molecule names> <options>
```

`<input.vcf>`

input coordinate filename (must end with `.vcf`) containing either ordered or indexed timesteps (with joined coordinates)

`<output.vcf>`

output filename with indexed coordinates (must end with `.vcf`)

`<molecule names>`

names of molecule types (linear chains) to use

Todo Possibly remove the necessity for joined coordinates.

4.2 PersistenceLength utility

This utility calculates persistence length of specified molecules. It is assumed that the provided molecules are linear chains, but no check is performed. Also the use of joined coordinates (that is, without periodic boundary condition) is required, because the utility does not remove periodic boundary conditions.

The calculation of the persistence length is based on the projection of angles between bonds vectors (see e.g. `this paper`). The following formula for the persistence length, l_{t} is used:

$$l_{\mathrm{P}} = \langle b \rangle \sum_{i=0}^{i=N_b} \langle \cos\theta_i \rangle, \tag{1}$$

where $\langle b \rangle$ is the average bond length in a molecule, $\langle \theta_i \rangle$ is the average angle between two bond vectors separated by i bonds. N_b is the number of bonds in the given molecule.

The output is a file containing average persistence length for every molecule type for each timestep.

Usage:

```
PersistenceLength <input.vcf> <output.vcf> <molecule names> <options>
```

`<input.vcf>`

input coordinate filename (must end with `.vcf`) containing either ordered or indexed timesteps (with joined coordinates)

`<output.vcf>`

output filename with indexed coordinates (must end with `.vcf`)

`<molecule names>`

names of molecule types (linear chains) to use

Todo Possibly remove the necessity for joined coordinates.

5 Utilities for nanoparticles

This section provides information about utilities used to handle simulations with nanoparticles.

5.1 C60 utility

This program creates a nanoparticle of a given radius. The inner part is formed by a C60 fullerene with diameter of 0.5 of the nanoparticle radius. The shell is formed by a C60 fullerene with added beads in the middle of every C60 face. There 153 beads in the nanoparticle.

Bonds are created between all beads, that is 153 $*$ 152 / 2 = 11628 bonds in all.

Two files are created: `C60.vtf` to check how the nanoparticle looks and `Nano-FIELD` with coordinates and bonds in the format required by the `FIELD` file in `DL_MESO simulation package`

Usage:

`C60 <radius>`

`<radius>`

required radius of nanoparticle (in reduced units)

The standard options cannot be used with this utility.

6 Todo List

Page Common utilities

Implement `<skip>` and `<start>`

Implement possibility to choose timestep number for creating `CONFIG` file.

Somehow avoid the need to use the special optional bond file, where the ids of beads must strictly adhere to FIELD. Possibly require use of a vcf file in conjunction with bond file

Add the possibility to save only certain bead types to output vcf file with joined coordinates.

Global ReadStructure (char $*$vsf_file, char $*$vcf_file, char $*$bonds_file, Counts $*$Counts, BeadType $**$Bead↩ Type, Bead $**$Bead, MoleculeType $**$MoleculeType, Molecule $**$Molecule)

Assigning Bead[].Index should be in auxiliary ReadVsf().

Page Utilities for linear chains

Possibly remove the necessity for joined coordinates.

Possibly remove the necessity for joined coordinates.

7 Data Structure Index

7.1 Data Structures

Here are the data structures with brief descriptions:

Aggregate
Information about every aggregate 11

Bead
Information about every bead 12

BeadType
Information about bead types 12

Counts
Total numbers of various things 13

Molecule
Information about every molecule 13

MoleculeType
Information about molecule types 13

Vector
3D vector 14

8 File Index

8.1 File List

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

AnalysisTools.h
Structures and functions common to all analysis utilities 14

Common/Aggregates.h ??

Common/TransformVsf.h ??

9 Data Structure Documentation

9.1 Aggregate Struct Reference

Information about every aggregate.

```
#include <AnalysisTools.h>
```

Data Fields

- int nMolecules

    number of molecules in aggregate
- int ∗ Molecule

    ids of molecules in aggregate
- int nBeads

    number of bonded beads in aggregate
- int ∗ Bead

    ids of bonded beads in aggregate
- int nMonomers

    number of monomeric beads in aggregate
- int ∗ Monomer

    ids of monomeric beads in aggregate

9.2 Bead Struct Reference

Information about every bead.

```
#include <AnalysisTools.h>
```

Data Fields

- int Type

    type of bead corresponding to index in BeadType struct
- int Molecule

    index number of molecule corresponding to Molecule struct (-1 for monomeric bead)
- int Index

    index of the bead according to .vsf file (needed for indexed timesteps)
- Vector Position

    cartesian coordinates of the bead

9.3 BeadType Struct Reference

Information about bead types.

```
#include <AnalysisTools.h>
```

Data Fields

- char Name [16]

    name of given bead type
- int Number

    number of beads of given type
- bool Use

    should bead type in .vcf file be used for calculation?
- bool Write

    should bead type in .vcf file be written to output .vcf?
- double Charge

    charge of every bead of given type
- double Mass

    mass of every bead of given type

9.4 Counts Struct Reference

Total numbers of various things.

```
#include <AnalysisTools.h>
```

Data Fields

- int TypesOfBeads

  number of bead types
- int TypesOfMolecules

  number of molecule types
- int Bonded

  total number of beads in all molecules
- int Unbonded

  total number of monomeric beads
- int Molecules

  total number of molecules
- int Aggregates

  total number of aggregates

9.5 Molecule Struct Reference

Information about every molecule.

```
#include <AnalysisTools.h>
```

Data Fields

- int Type

  type of molecule corresponding to index in MoleculeType struct
- int ∗ Bead

  ids of beads in the molecule

9.6 MoleculeType Struct Reference

Information about molecule types.

```
#include <AnalysisTools.h>
```

Data Fields

- char Name [16]

    name of given molecule type
- int Number

    number of molecules of given type
- int nBeads

    number of beads in every molecule of given type
- int nBonds

    number of bonds in every molecule of given type
- int ∗∗ Bond

    pair of ids for every bond (with relative bead numbers from 0 to nBeads)
- double Mass

    total mass of every molecule of given type
- bool Use

    should molecule type be used for calculation?

9.7 Vector Struct Reference

3D vector.

```
#include <AnalysisTools.h>
```

Data Fields

- double **x**
- double **y**
- double **z**

10 File Documentation

10.1 AnalysisTools.h File Reference

Structures and functions common to all analysis utilities.

Data Structures

- struct Vector

    3D vector.
- struct Counts

    Total numbers of various things.
- struct BeadType

    Information about bead types.
- struct MoleculeType

    Information about molecule types.
- struct Bead

    Information about every bead.
- struct Molecule

    Information about every molecule.
- struct Aggregate

    Information about every aggregate.

Macros

- #define SQR(x) ((x)∗(x))

  macro for algebraic square
- #define CUBE(x) ((x)∗(x)∗(x))

  macro for algebraic cube

Typedefs

- typedef struct Vector Vector

  3D vector.
- typedef struct Counts Counts

  Total numbers of various things.
- typedef struct BeadType BeadType

  Information about bead types.
- typedef struct MoleculeType MoleculeType

  Information about molecule types.
- typedef struct Bead Bead

  Information about every bead.
- typedef struct Molecule Molecule

  Information about every molecule.
- typedef struct Aggregate Aggregate

  Information about every aggregate.

Functions

- bool ReadStructure (char ∗vsf_file, char ∗vcf_file, char ∗bonds_file, Counts ∗Counts, BeadType ∗∗BeadType, Bead ∗∗Bead, MoleculeType ∗∗MoleculeType, Molecule ∗∗Molecule)

  Function reading information from dl_meso FIELD and vsf structure files.
- int ReadCoorOrdered (FILE ∗vcf_file, Counts Counts, Bead ∗∗Bead, char ∗∗stuff)

  Function reading ordered coordinates from .vcf coordinate file.
- int ReadCoorIndexed (FILE ∗vcf_file, Counts Counts, Bead ∗∗Bead, char ∗∗stuff)

  Function reading ordered coordinates from .vcf coordinate file.
- void VerboseOutput (char ∗input_vcf, char ∗bonds_file, Counts Counts, BeadType ∗BeadType, Bead ∗Bead, MoleculeType ∗MoleculeType, Molecule ∗Molecule)

  Function printing basic information about system if $-v$ or $-V$ option is provided.
- void WriteCoorIndexed (FILE ∗vcf_file, Counts Counts, BeadType ∗BeadType, Bead ∗Bead, char ∗stuff)

  Function writing indexed coordinates to a .vcf file.
- int FindBeadType (char ∗name, Counts Counts, BeadType ∗BeadType)

  Function to identify type of bead from its name.
- int FindMoleculeType (char ∗name, Counts Counts, MoleculeType ∗MoleculeType)

  Function to identify type of bead from its name.
- Vector DistanceBetweenBeads (int id1, int id2, Bead ∗Bead, Vector BoxLength)

  Function to calculate distance vector between two beads.
- void FillAggregateBeads (Aggregate ∗∗Aggregate, Counts Counts, MoleculeType ∗MoleculeType, Molecule ∗Molecule)

  Function to assign bead ids to Aggregate struct.
- void RemovePBCMolecules (Counts Counts, Vector BoxLength, BeadType ∗BeadType, Bead ∗∗Bead, MoleculeType ∗MoleculeType, Molecule ∗Molecule)

  Function to join all molecules.

10.1.1 Function Documentation

10.1.1.1 Vector DistanceBetweenBeads (int *id1,* int *id2,* Bead ∗ *Bead,* Vector *BoxLength*)

Parameters

in	*id1*	index of first bead for distance calculation
in	*id2*	index of second bead for distance calculation
in	*Bead*	information about individual beads (i.e. coordinates)
in	*BoxLength*	dimensions of simulation box

Returns

distance vector between the two provided beads (without pbc)

Function calculating distance vector between two beads. It removes periodic boundary conditions and returns x, y, and z distances in the range <0, BoxLength/2).

10.1.1.2 void FillAggregateBeads (Aggregate ∗∗ *Aggregate*, Counts *Counts*, MoleculeType ∗ *MoleculeType*, Molecule ∗ *Molecule*)

Parameters

out	*Aggregate*	information about all aggregates (i.e. bead ids in aggregates)
in	*Counts*	numbers of beads, molecules, etc.
in	*MoleculeType*	information about molecule types
in	*Molecule*	information about individual molecules

Function to assign bead ids to aggregates accoding to molecules in the aggregates. It essentially duplicates the information for the convenience of easy access to all beads in the aggregates.

10.1.1.3 int FindBeadType (char ∗ *name*, Counts *Counts*, BeadType ∗ *BeadType*)

Parameters

in	*name*	bead name
in	*Counts*	numbers of beads, residues, etc.
in	*BeadType*	informationn about bead types

Returns

bead type id corresponding to index in BeadType struct

10.1.1.4 int FindMoleculeType (char ∗ *name*, Counts *Counts*, MoleculeType ∗ *MoleculeType*)

Parameters

in	*name*	bead name
in	*Counts*	numbers of beads, residues, etc.
in	*MoleculeType*	informationn about bead types

Returns

molecule type id corresponding to index in BeadType struct

10.1.1.5 int ReadCoorIndexed (FILE ∗ *vcf_file*, Counts *Counts*, Bead ∗∗ *Bead*, char ∗∗ *stuff*)

Parameters

`in`	*vcf_file*	name of input .vcf coordinate file
`in`	*Counts*	numbers of beads, molecules, etc.
`out`	*Bead*	coordinates of individual beads
`out`	*stuff*	first line of a timestep

Returns

0 for no errors or index number of bead (starting from 1) for which coordinates cannot be read

Function reading coordinates from .vcf file with indexed timesteps (Indexed coordinate file).

10.1.1.6 int ReadCoorOrdered (FILE ∗ *vcf_file*, Counts *Counts*, Bead ∗∗ *Bead*, char ∗∗ *stuff*)

Parameters

`in`	*vcf_file*	name of input .vcf coordinate file
`in`	*Counts*	numbers of beads, molecules, etc.
`out`	*Bead*	coordinates of individual beads
`out`	*stuff*	first line of a timestep

Returns

0 for no errors or index number of bead (starting from 1) for which coordinates cannot be read

Function reading coordinates from .vcf file with ordered timesteps (Ordered coordinate file).

10.1.1.7 bool ReadStructure (char ∗ *vsf_file*, char ∗ *vcf_file*, char ∗ *bonds_file*, Counts ∗ *Counts*, BeadType ∗∗ *BeadType*, Bead ∗∗ *Bead*, MoleculeType ∗∗ *MoleculeType*, Molecule ∗∗ *Molecule*)

Parameters

`in`	*vsf_file*	.vsf structure file
`in`	*vcf_file*	.vcf coordinate file
`in`	*bonds_file*	filename with bonds
`out`	*Counts*	numbers of beads, molecules, etc.
`out`	*BeadType*	information about bead types
`out`	*Bead*	informationn about individual beads
`out`	*MoleculeType*	information about molecule types
`out`	*Molecule*	information about individual molecules

Returns

'true' or 'false' for .vcf file with indexed or ordered timesteps, respectively

Function reading information about beads and molecules from DL_MESO `FIELD` file and a `.vsf` structure file. Name, mass and charge of every bead type is read from `species` lines in `FIELD`. The number of molecule types are read from `molecule` section. For each molecule type its name, the number of molecules, the number of beads and bonds in each molecule and the bonds themselves are read. Input structure file provides information about what bead is of which type. Optional file with bond declarations provides an alternative for bonds of any molecule type in `FIELD`. If optional bond file is not used, an empty string is passed to this function.

Todo Assigning Bead[].Index should be in auxiliary ReadVsf().

10.1.1.8 void RemovePBCMolecules (**Counts** *Counts,* **Vector** *BoxLength,* **BeadType** ∗ *BeadType,* **Bead** ∗∗ *Bead,* **MoleculeType** ∗ *MoleculeType,* **Molecule** ∗ *Molecule*)

Parameters

`in`	*Counts*	numbers of beads, molecules, etc.
`in`	*BoxLength*	dimension of the simulation box
`in`	*BeadType*	information about bead types
`out`	*Bead*	information about individual beads (coordinates)
`in`	*MoleculeType*	information about molecule types
`in`	*Molecule*	information about individual molecules

Function to remove periodic boundary conditions from all individual molecules, thus joining them

10.1.1.9 void VerboseOutput (char ∗ *input_vcf,* char ∗ *bonds_file,* **Counts** *Counts,* **BeadType** ∗ *BeadType,* **Bead** ∗ *Bead,* **MoleculeType** ∗ *MoleculeType,* **Molecule** ∗ *Molecule*)

Parameters

`in`	*input_vcf*	.vcf structure file
`in`	*bonds_file*	filename with bonds
`in`	*Counts*	numbers of beads, molecules, etc.
`in`	*BeadType*	information about bead types
`in`	*Bead*	informationn about individual beads
`in`	*MoleculeType*	information about molecule types
`in`	*Molecule*	information about individual molecules

10.1.1.10 void WriteCoorIndexed (FILE ∗ *vcf_file,* **Counts** *Counts,* **BeadType** ∗ *BeadType,* **Bead** ∗ *Bead,* char ∗ *stuff*)

Parameters

`in`	*vcf_file*	name of output .vcf coordinate file
`in`	*Counts*	numbers of beads, molecules, etc.
`in`	*BeadType*	information about bead types
`in`	*Bead*	coordinates of individual beads
`in`	*stuff*	array of chars containing comment line to place at the beginning

Function writing coordinates to a `.vcf` file. According to the Use flag in BeadType structure only certain bead types will be saved into the indexed timestep in .vcf file (Indexed coordinate file).