Project documentation

Generated by Doxygen 1.9.1

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

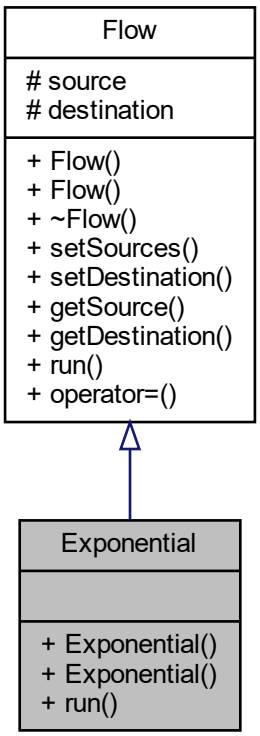
Here is a list of all files with brief descriptions:

4 Class Documentation

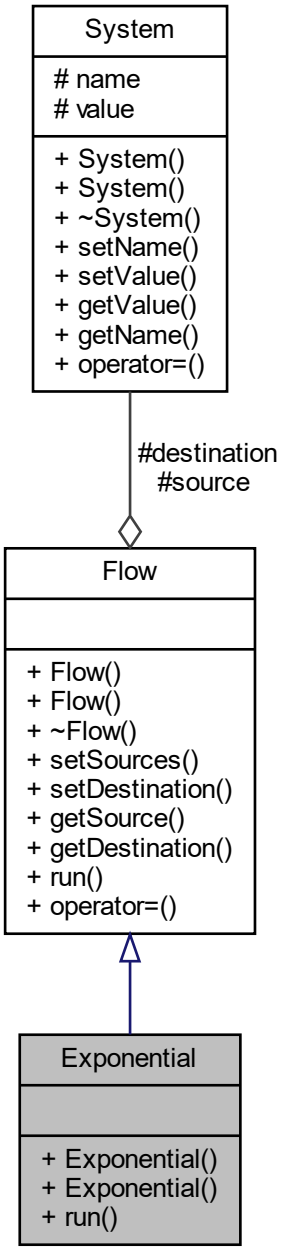
4.1 Exponential Class Reference

`#include <flow.h>`

Inheritance diagram for Exponential:



Collaboration diagram for Exponential:



Public Member Functions

- Exponential ()

  Builder to create a new exponential flow.
- Exponential (System ∗source, System ∗destination)
- double run ()

  Function to run the stream.

Additional Inherited Members

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Exponential() [1/2] Exponential::Exponential ( )

Builder to create a new exponential flow.

4.1.1.2 Exponential() [2/2] Exponential::Exponential (

```
System * source,
System * destination )  [inline]
```

4.1.2 Member Function Documentation

4.1.2.1 run() double Exponential::run ( ) [inline], [virtual]

Function to run the stream.

Returns

Returns double resulting from calculation performed.

Implements Flow.

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

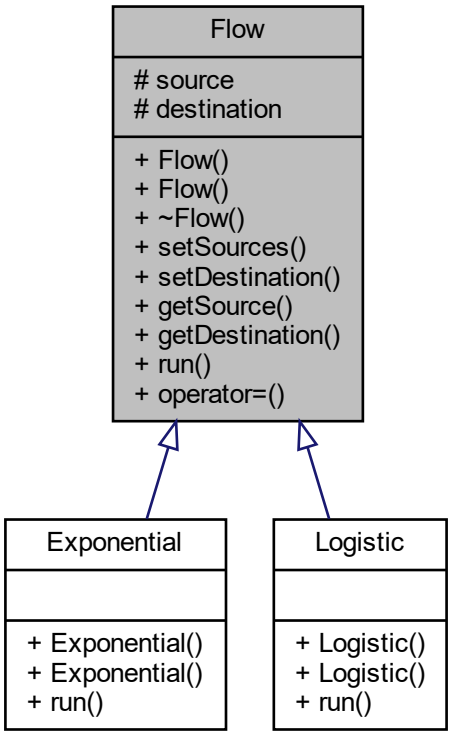
- MyVensim/src/flow.h

4.2 Flow Class Reference

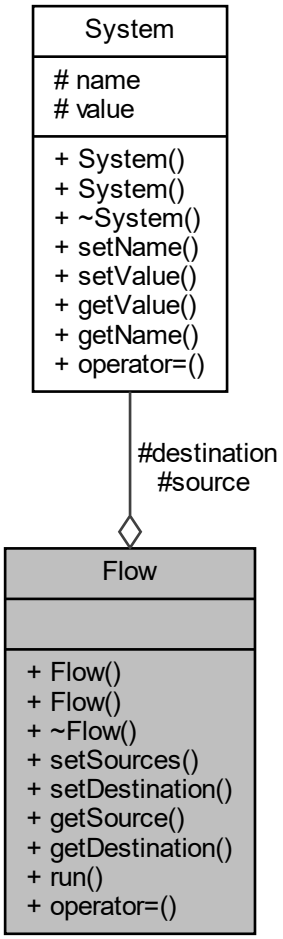
File responsible for project flows.

`#include <flow.h>`

Inheritance diagram for Flow:



Collaboration diagram for Flow:



Public Member Functions

- Flow ()

  Builder to create a new stream.
- Flow (System ∗, System ∗)
- virtual ∼Flow ()

  Destructor to destroy the flow.
- void setSources (System ∗)

  Add an input system to the stream.
- void setDestination (System ∗)

  Add an exit system to the stream.
- System ∗ getSource ()

  Function to return an input system.
- System ∗ getDestination ()

  Function to return an output system.

- virtual double run ()=0

  Virtual function to run the stream.
- Flow * operator= (Flow *)

  Function to overload operator =.

Protected Attributes

- System * source
- System * destination

4.2.1 Detailed Description

File responsible for project flows.

Author

Ananda Mendes 2021.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Flow() [1/2] Flow::Flow ( )

Builder to create a new stream.

<Pointer of output of a system

4.2.2.2 Flow() [2/2]

```
Flow::Flow (
        System * source,
        System * destination )
```

4.2.2.3 ~Flow() Flow::~Flow ( ) [virtual]

Destructor to destroy the flow.

4.2.3 Member Function Documentation

4.2.3.1 getDestination() `System * Flow::getDestination ( )`

Function to return an output system.

Returns

Returns a System object.

4.2.3.2 getSource() `System * Flow::getSource ( )`

Function to return an input system.

Returns

Returns a System object.

4.2.3.3 operator=() `Flow * Flow::operator= ( Flow * flow )`

Function to overload operator =.

Parameters

flow	Flow pointer.

Returns

Returns flow.

4.2.3.4 run() `virtual double Flow::run ( ) [pure virtual]`

Virtual function to run the stream.

Returns

Returns value of 0.

Implemented in Logistic, and Exponential.

4.2.3.5 setDestination() `void Flow::setDestination ( System * destination )`

Add an exit system to the stream.

Parameters

system	System pointer.

4.2.3.6 setSources() `void Flow::setSources ( System * source )`

Add an input system to the stream.

Parameters

system	System pointer.

4.2.4 Member Data Documentation

4.2.4.1 destination `System* Flow::destination [protected]`

<Pointer of entry of a system

4.2.4.2 source `System* Flow::source [protected]`

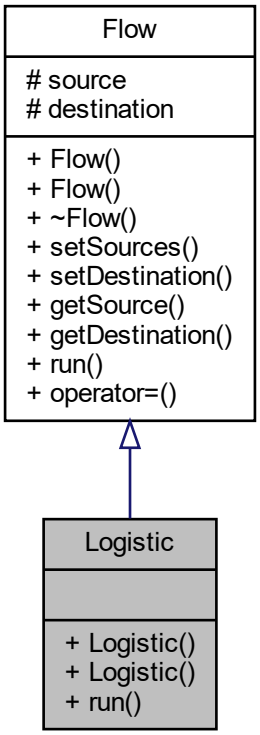
The documentation for this class was generated from the following files:

- MyVensim/src/flow.h
- MyVensim/src/flow.cpp

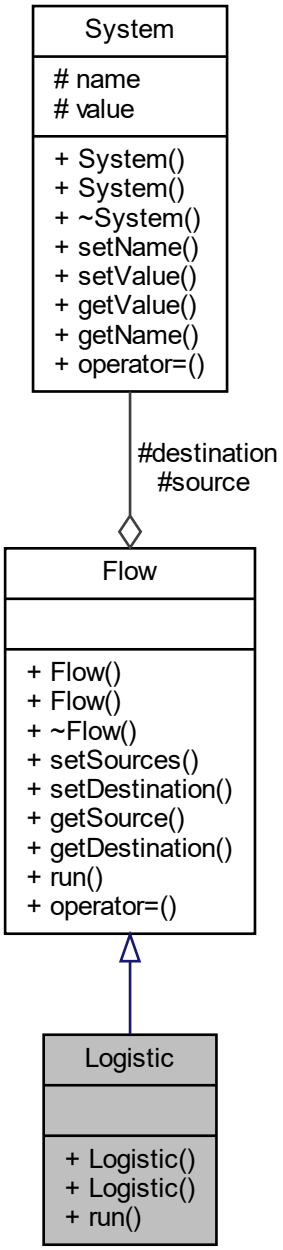
4.3 Logistic Class Reference

`#include <flow.h>`

Inheritance diagram for Logistic:



Collaboration diagram for Logistic:



Public Member Functions

- Logistic ()

  Builder to create a new logistical flow.
- Logistic (System ∗source, System ∗destination)
- double run ()

  Function to run the stream.

Additional Inherited Members

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Logistic() [1/2] Logistic::Logistic ( )

Builder to create a new logistical flow.

4.3.1.2 Logistic() [2/2] Logistic::Logistic (System * source, System * destination) [inline]

4.3.2 Member Function Documentation

4.3.2.1 run() double Logistic::run ( ) [inline], [virtual]

Function to run the stream.

Returns

Returns double resulting from calculation performed.

Implements Flow.

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

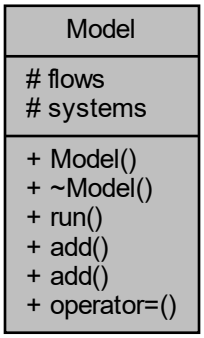
- MyVensim/src/flow.h

4.4 Model Class Reference

File responsible for project templates.

`#include <model.h>`

Collaboration diagram for Model:



Public Member Functions

- Model ()

  Builder to create a new model.
- virtual ~Model ()

  Destructor to destroy the model.
- double run (int, int)

  Function to run the model.
- void add (System *)

  Function to add a new system.
- void add (Flow *)

  Function to add new flow.
- Model * operator= (Model *)

  Function to overload operator =.

Protected Attributes

- vector< Flow * > flows
- vector< System * > systems

4.4.1 Detailed Description

File responsible for project templates.

Author

Ananda Mendes 2021.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Model() `Model::Model ( )`

Builder to create a new model.

< Systems pointer vector

4.4.2.2 ~Model() `Model::~Model ( ) [virtual]`

Destructor to destroy the model.

4.4.3 Member Function Documentation

4.4.3.1 add() [1/2] void Model::add (Flow * flow)

Function to add new flow.

4.4.3.2 add() [2/2] void Model::add (System * system)

Function to add a new system.

4.4.3.3 operator=() Model * Model::operator= (Model * model)

Function to overload operator =.

Parameters

model	Model pointer.

Returns

Returns model.

4.4.3.4 run() double Model::run (int start, int finish)

Function to run the model.

Parameters

start	Initial value.
finish	Final value.

Returns

Returns final value.

4.4.4 Member Data Documentation

4.4.4.1 flows `vector<Flow*> Model::flows [protected]`

4.4.4.2 systems `vector<System*> Model::systems [protected]`

< Flow pointer vector

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

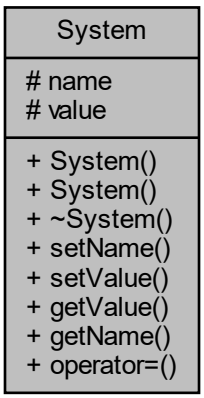
- MyVensim/src/model.h
- MyVensim/src/model.cpp

4.5 System Class Reference

File responsible for project systems.

`#include <system.h>`

Collaboration diagram for System:



Public Member Functions

- System ()
  Builder to create a new system.
- System (string, double)
- virtual ~System ()
  Destructor to destroy the system.
- void setName (string)
  Add a name for the system.
- void setValue (double)
  Add a value to the system.
- double getValue ()
  Function to return system value.
- string getName ()
  Function to return system name.
- System * operator= (System *)
  Function to overload operator =.

Protected Attributes

- string name
- double value

4.5.1 Detailed Description

File responsible for project systems.

Author

Ananda Mendes 2021.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 System() [1/2] System::System ( )

Builder to create a new system.

<Double value

4.5.2.2 System() [2/2] System::System (string *name*, double *value*)

4.5.2.3 ~System() System::~System ( ) [virtual]

Destructor to destroy the system.

4.5.3 Member Function Documentation

4.5.3.1 getName() string System::getName ( )

Function to return system name.

Returns

Returns a string.

4.5.3.2 getValue() double System::getValue ( )

Function to return system value.

Returns

Returns a double.

4.5.3.3 operator=() System * System::operator= (System * *system*)

Function to overload operator =.

Parameters

flow	System pointer.

Returns

Returns a system.

4.5.3.4 setName() `void System::setName ( string name )`

Add a name for the system.

Parameters

name	System name.

4.5.3.5 setValue() `void System::setValue ( double value )`

Add a value to the system.

Parameters

value	System value.

4.5.4 Member Data Documentation

4.5.4.1 name `string System::name [protected]`

4.5.4.2 value `double System::value [protected]`

<String name

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

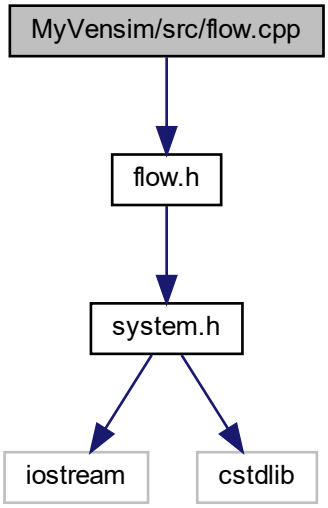
- MyVensim/src/system.h
- MyVensim/src/system.cpp

5 File Documentation

5.1 MyVensim/src/flow.cpp File Reference

```
#include "flow.h"
```

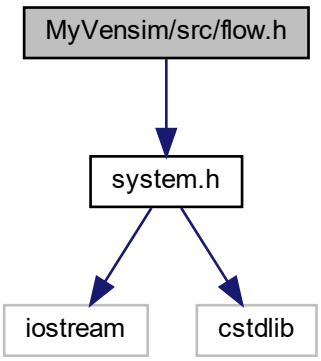
Include dependency graph for flow.cpp:



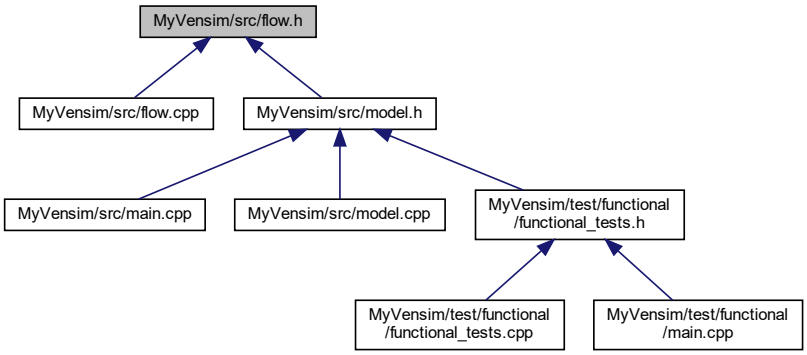
5.2 MyVensim/src/flow.h File Reference

```
#include "system.h"
```

Include dependency graph for flow.h:



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



Classes

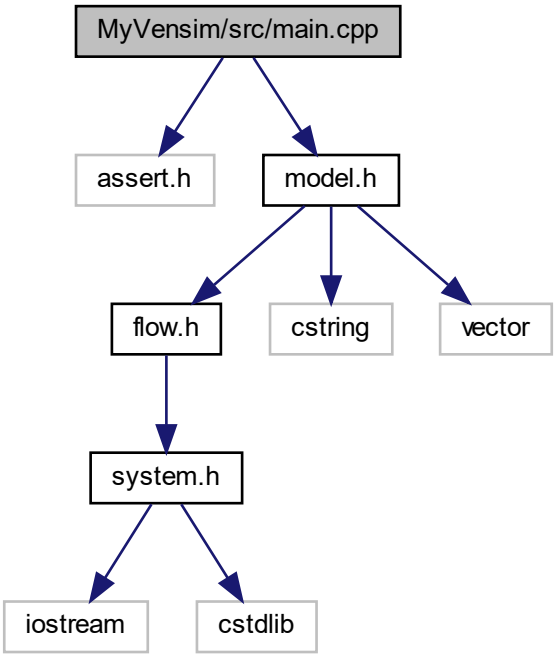
- class Flow

  File responsible for project flows.
- class Exponential
- class Logistic

5.3 MyVensim/src/main.cpp File Reference

```
#include <assert.h>
#include "model.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

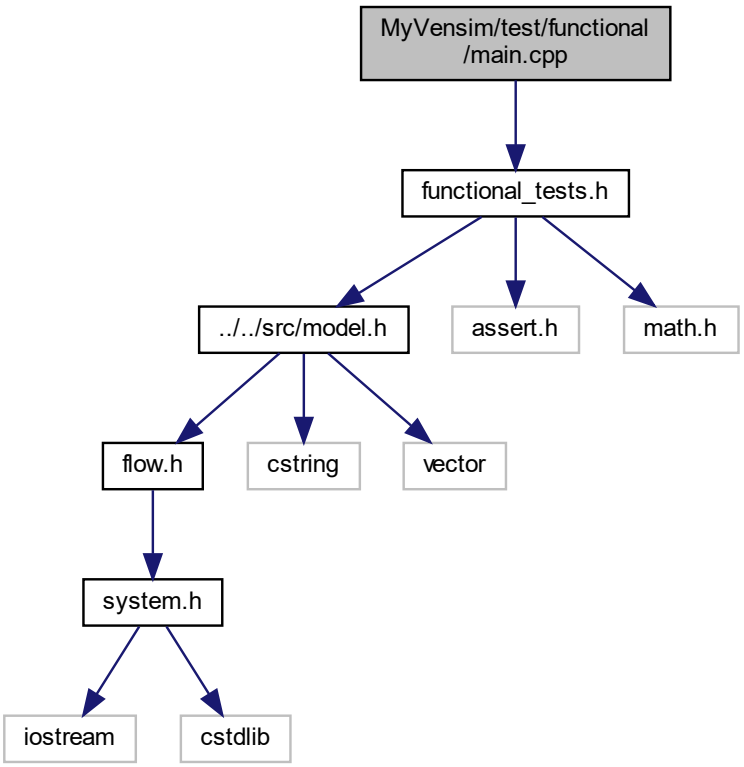
5.3.1 Function Documentation

5.3.1.1 main() `int main ( )`

5.4 MyVensim/test/functional/main.cpp File Reference

```
#include "functional_tests.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

5.4.1 Function Documentation

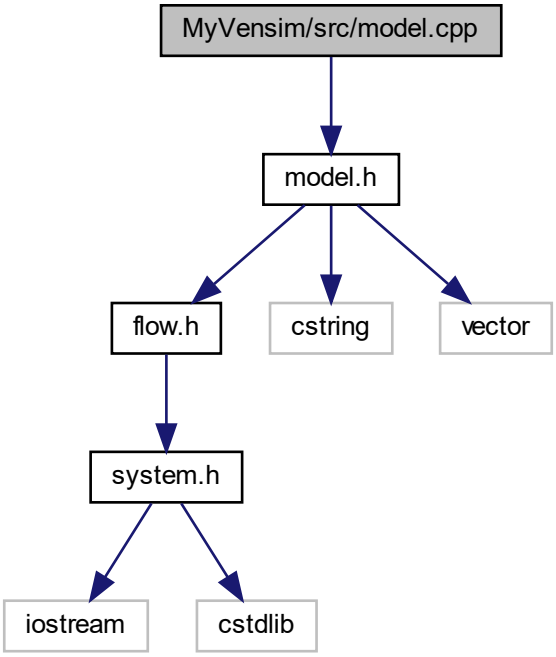
5.4.1.1 main() `int main ( )`

5.5 MyVensim/test/unit/main.cpp File Reference

5.6 MyVensim/src/model.cpp File Reference

```
#include "model.h"
```

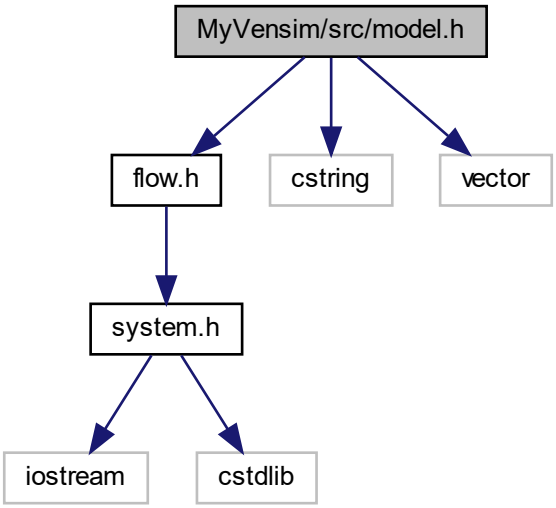
Include dependency graph for model.cpp:



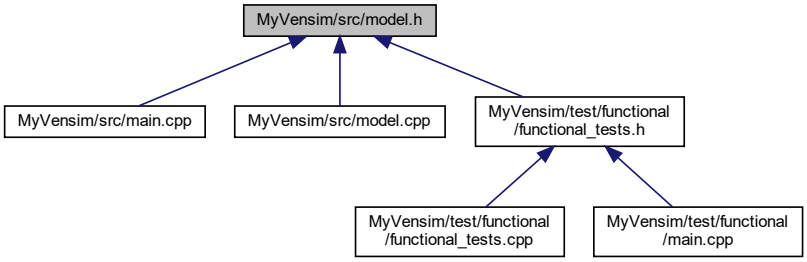
5.7 MyVensim/src/model.h File Reference

```
#include "flow.h"
#include <cstring>
#include <vector>
```

Include dependency graph for model.h:



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



Classes

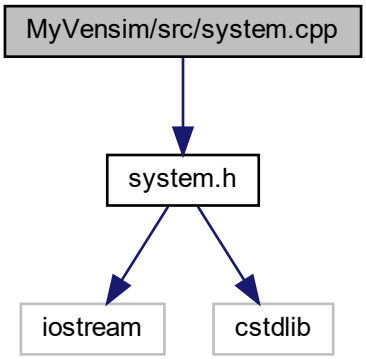
- class Model

  File responsible for project templates.

5.8 MyVensim/src/system.cpp File Reference

```
#include "system.h"
```

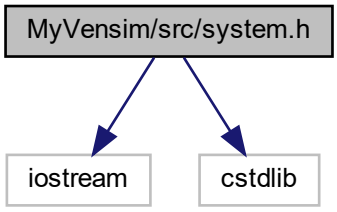
Include dependency graph for system.cpp:



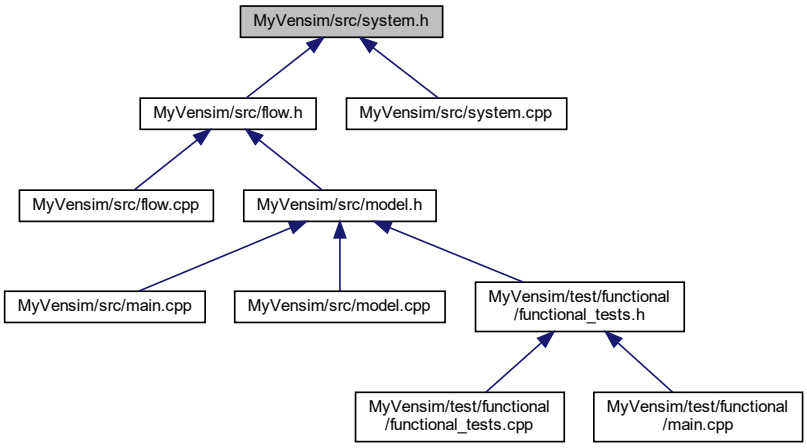
5.9 MyVensim/src/system.h File Reference

```
#include <iostream>
#include <cstdlib>
```

Include dependency graph for system.h:



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



Classes

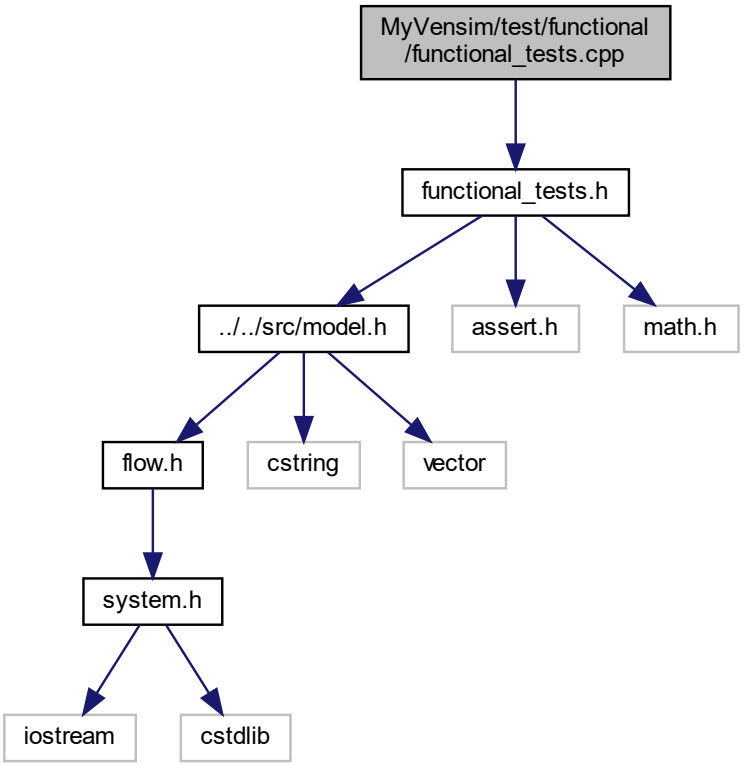
- class System

  File responsible for project systems.

5.10 MyVensim/test/functional/functional_tests.cpp File Reference

```
#include "functional_tests.h"
```

Include dependency graph for functional_tests.cpp:



Functions

- void exponentialFuncionalTest ()

  File responsible for functional testing.
- void logisticalFuncionalTest ()

  Logistics functional test.
- void complexFuncionalTest ()

  Complex functional test.

5.10.1 Function Documentation

5.10.1.1 complexFuncionalTest() `void complexFuncionalTest ( )`

Complex functional test.

5.10.1.2 exponentialFuncionalTest() `void exponentialFuncionalTest ( )`

File responsible for functional testing.

Author

Ananda Mendes 2021.

Exponential functional test.

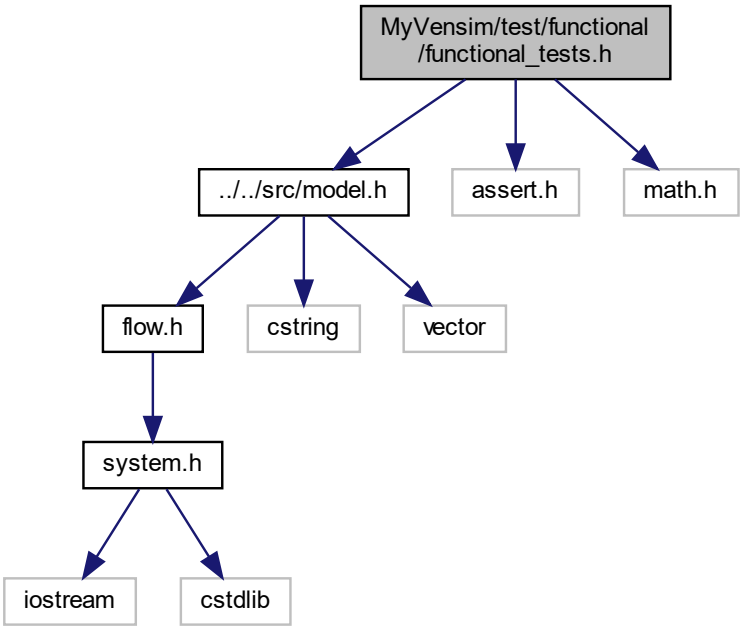
5.10.1.3 logisticalFuncionalTest() `void logisticalFuncionalTest ( )`

Logistics functional test.

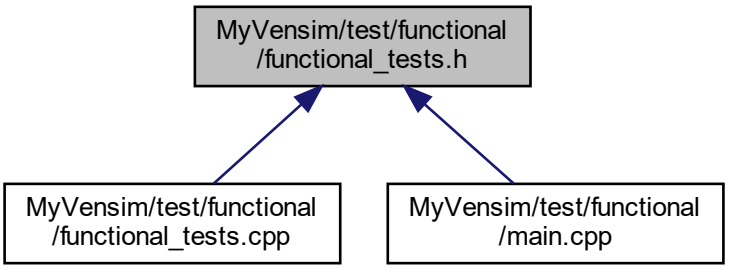
5.11 MyVensim/test/functional/functional_tests.h File Reference

```
#include "../../src/model.h"
#include <assert.h>
#include <math.h>
```

Include dependency graph for functional_tests.h:



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



Functions

- void exponentialFuncionalTest ()
  File responsible for functional testing.
- void logisticalFuncionalTest ()
  Logistics functional test.
- void complexFuncionalTest ()
  Complex functional test.

5.11.1 Function Documentation

5.11.1.1 complexFuncionalTest() `void complexFuncionalTest ( )`

Complex functional test.

5.11.1.2 exponentialFuncionalTest() `void exponentialFuncionalTest ( )`

File responsible for functional testing.

Author

Ananda Mendes 2021.

Exponential functional test.

5.11.1.3 logisticalFuncionalTest() `void logisticalFuncionalTest ( )`

Logistics functional test.

5.12 MyVensim/test/unit/unit_tests.cpp File Reference

5.13 MyVensim/test/unit/unit_tests.h File Reference