

MyVensin

Generated by Doxygen 1.9.3



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Exponential Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 Exponential() [1/2]	8
4.1.2.2 Exponential() [2/2]	9
4.1.3 Member Function Documentation	9
4.1.3.1 run()	9
4.2 Flow Class Reference	9
4.2.1 Detailed Description	10
4.2.2 Constructor & Destructor Documentation	10
4.2.2.1 ~Flow()	10
4.2.3 Member Function Documentation	10
4.2.3.1 getDestination()	11
4.2.3.2 getSource()	11
4.2.3.3 operator=()	11
4.2.3.4 run()	12
4.2.3.5 setDestination()	12
4.2.3.6 setSources()	12
4.3 Flow_Imp Class Reference	12
4.3.1 Detailed Description	14
4.3.2 Constructor & Destructor Documentation	14
4.3.2.1 Flow_Imp() [1/2]	14
4.3.2.2 Flow_Imp() [2/2]	14
4.3.2.3 ~Flow_Imp()	15
4.3.3 Member Function Documentation	15
4.3.3.1 getDestination()	15
4.3.3.2 getSource()	15
4.3.3.3 operator=()	15
4.3.3.4 run()	16
4.3.3.5 setDestination()	16
4.3.3.6 setSources()	17
4.3.4 Member Data Documentation	17
4.3.4.1 destination	17

4.3.4.2 source	17
4.4 FlowUnit Class Reference	18
4.4.1 Detailed Description	18
4.5 Logistic Class Reference	19
4.5.1 Detailed Description	20
4.5.2 Constructor & Destructor Documentation	20
4.5.2.1 Logistic() [1/2]	20
4.5.2.2 Logistic() [2/2]	20
4.5.3 Member Function Documentation	20
4.5.3.1 run()	21
4.6 Model Class Reference	21
4.6.1 Detailed Description	22
4.6.2 Constructor & Destructor Documentation	22
4.6.2.1 ~Model()	22
4.6.3 Member Function Documentation	22
4.6.3.1 createFlow()	22
4.6.3.2 createModel()	22
4.6.3.3 createSystem()	23
4.6.3.4 getFlows()	23
4.6.3.5 getSystem()	23
4.6.3.6 operator=()	24
4.6.3.7 run()	24
4.7 Model_Imp Class Reference	24
4.7.1 Detailed Description	25
4.7.2 Constructor & Destructor Documentation	26
4.7.2.1 Model_Imp() [1/2]	26
4.7.2.2 Model_Imp() [2/2]	26
4.7.2.3 ~Model_Imp()	26
4.7.3 Member Function Documentation	26
4.7.3.1 createModel()	26
4.7.3.2 createSystem()	27
4.7.3.3 getFlows()	27
4.7.3.4 getSystem()	27
4.7.3.5 operator=()	27
4.7.3.6 run()	28
4.7.4 Member Data Documentation	28
4.7.4.1 flows	28
4.7.4.2 models	28
4.7.4.3 systems	28
4.8 System Class Reference	29
4.8.1 Detailed Description	29
4.8.2 Constructor & Destructor Documentation	29

4.8.2.1 ~System()	30
4.8.3 Member Function Documentation	30
4.8.3.1 getName()	30
4.8.3.2 getValue()	30
4.8.3.3 operator=()	30
4.8.3.4 setName()	31
4.8.3.5 setValue()	31
4.9 System_Imp Class Reference	31
4.9.1 Detailed Description	33
4.9.2 Constructor & Destructor Documentation	33
4.9.2.1 System_Imp() [1/2]	33
4.9.2.2 System_Imp() [2/2]	33
4.9.2.3 ~System_Imp()	33
4.9.3 Member Function Documentation	34
4.9.3.1 getName()	34
4.9.3.2 getValue()	34
4.9.3.3 operator=()	34
4.9.3.4 setName()	35
4.9.3.5 setValue()	35
4.9.4 Member Data Documentation	35
4.9.4.1 name	35
4.9.4.2 value	36
<b>5 File Documentation</b>	<b>37</b>
5.1 src/lib/flow.h File Reference	37
5.2 flow.h	38
5.3 src/lib/flow_Imp.cpp File Reference	39
5.4 flow_Imp.cpp	39
5.5 src/lib/flow_Imp.h File Reference	40
5.6 flow_Imp.h	41
5.7 src/lib/flow_unit.h File Reference	42
5.8 flow_unit.h	43
5.9 src/lib/model.h File Reference	43
5.10 model.h	44
5.11 src/lib/model_Imp.cpp File Reference	45
5.12 model_Imp.cpp	45
5.13 src/lib/model_Imp.h File Reference	47
5.14 model_Imp.h	48
5.15 src/lib/mySim.cpp File Reference	49
5.16 mySim.cpp	49
5.17 src/lib/mySlim.h File Reference	49
5.18 mySlim.h	49

5.19 src/lib/system.h File Reference	49
5.20 system.h	50
5.21 src/lib/system_Imp.cpp File Reference	50
5.22 system_Imp.cpp	51
5.23 src/lib/system_Imp.h File Reference	51
5.24 system_Imp.h	52
5.25 src/main.cpp File Reference	52
5.25.1 Function Documentation	53
5.25.1.1 main()	53
5.26 main.cpp	53
5.27 test/functional/main.cpp File Reference	54
5.27.1 Function Documentation	54
5.27.1.1 main()	54
5.28 main.cpp	55
5.29 test/unit/main.cpp File Reference	55
5.29.1 Function Documentation	55
5.29.1.1 main()	56
5.30 main.cpp	56
5.31 test/functional/functional_tests.cpp File Reference	56
5.31.1 Function Documentation	57
5.31.1.1 complexFuncionalTest()	57
5.31.1.2 exponentialFuncionalTest()	58
5.31.1.3 logisticalFuncionalTest()	58
5.32 functional_tests.cpp	58
5.33 test/functional/functional_tests.h File Reference	59
5.33.1 Function Documentation	60
5.33.1.1 complexFuncionalTest()	60
5.33.1.2 exponentialFuncionalTest()	60
5.33.1.3 logisticalFuncionalTest()	61
5.34 functional_tests.h	61
5.35 test/unit/mem_usage.cpp File Reference	61
5.35.1 Function Documentation	62
5.35.1.1 mem_usage()	62
5.36 mem_usage.cpp	62
5.37 test/unit/mem_usage.h File Reference	62
5.37.1 Function Documentation	63
5.37.1.1 mem_usage()	63
5.38 mem_usage.h	63
5.39 test/unit/unit_flow.cpp File Reference	64
5.39.1 Function Documentation	64
5.39.1.1 run_unit_test_Flow()	65
5.39.1.2 unit_Flow_constructor()	65

5.39.1.3 unit_Flow_destructor()	65
5.39.1.4 unit_Flow_getDestination()	65
5.39.1.5 unit_Flow_getSource()	65
5.39.1.6 unit_Flow_operator()	65
5.39.1.7 unit_Flow_setDestination()	66
5.39.1.8 unit_Flow_setSource()	66
5.40 unit_flow.cpp	66
5.41 test/unit/unit_flow.h File Reference	67
5.41.1 Function Documentation	68
5.41.1.1 run_unit_test_Flow()	68
5.41.1.2 unit_Flow_constructor()	68
5.41.1.3 unit_Flow_destructor()	69
5.41.1.4 unit_Flow_getDestination()	69
5.41.1.5 unit_Flow_getSource()	69
5.41.1.6 unit_Flow_operator()	69
5.41.1.7 unit_Flow_setDestination()	69
5.41.1.8 unit_Flow_setSource()	69
5.42 unit_flow.h	70
5.43 test/unit/unit_model.cpp File Reference	70
5.43.1 Function Documentation	71
5.43.1.1 run_unit_test_Model()	71
5.43.1.2 unit_Model_add_Flow()	71
5.43.1.3 unit_Model_add_System()	71
5.43.1.4 unit_Model_constructor()	71
5.43.1.5 unit_Model_destructor()	72
5.43.1.6 unit_Model_run()	72
5.44 unit_model.cpp	72
5.45 test/unit/unit_model.h File Reference	73
5.45.1 Function Documentation	74
5.45.1.1 run_unit_test_Model()	74
5.45.1.2 unit_Model_add_Flow()	74
5.45.1.3 unit_Model_add_System()	74
5.45.1.4 unit_Model_constructor()	74
5.45.1.5 unit_Model_destructor()	75
5.45.1.6 unit_Model_run()	75
5.46 unit_model.h	75
5.47 test/unit/unit_system.cpp File Reference	76
5.47.1 Function Documentation	76
5.47.1.1 run_unit_test_System()	76
5.47.1.2 unit_System_constructor()	77
5.47.1.3 unit_System_destructor()	77
5.47.1.4 unit_System_getName()	77

5.47.1.5 unit_System_getValue()	77
5.47.1.6 unit_System_operator()	77
5.47.1.7 unit_System_setName()	77
5.47.1.8 unit_System_setValue()	78
5.48 unit_system.cpp	78
5.49 test/unit/unit_system.h File Reference	79
5.49.1 Function Documentation	80
5.49.1.1 run_unit_test_System()	80
5.49.1.2 unit_System_constructor()	80
5.49.1.3 unit_System_destructor()	80
5.49.1.4 unit_System_getName()	80
5.49.1.5 unit_System_getValue()	80
5.49.1.6 unit_System_operator()	81
5.49.1.7 unit_System_setName()	81
5.49.1.8 unit_System_setValue()	81
5.50 unit_system.h	81
5.51 test/unit/unit_tests.cpp File Reference	82
5.52 unit_tests.cpp	82
5.53 test/unit/unit_tests.h File Reference	82
5.54 unit_tests.h	83
<b>Index</b>	<b>85</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Flow . . . . .	9
Flow_Imp . . . . .	12
Exponential . . . . .	7
FlowUnit . . . . .	18
Logistic . . . . .	19
Model . . . . .	21
Model_Imp . . . . .	24
System . . . . .	29
System_Imp . . . . .	31



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Exponential</a>	7
<a href="#">Flow</a>	
File responsible for project flows	9
<a href="#">Flow_Imp</a>	
File responsible for project flows	12
<a href="#">FlowUnit</a>	
File responsible for project flows	18
<a href="#">Logistic</a>	19
<a href="#">Model</a>	
File responsible for project templates	21
<a href="#">Model_Imp</a>	
File responsible for project templates	24
<a href="#">System</a>	
File responsible for project systems	29
<a href="#">System_Imp</a>	
File responsible for project systems	31



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

src/main.cpp	52
src/lib/flow.h	37
src/lib/flow_imp.cpp	39
src/lib/flow_imp.h	40
src/lib/flow_unit.h	42
src/lib/model.h	43
src/lib/model_imp.cpp	45
src/lib/model_imp.h	47
src/lib/mySim.cpp	49
src/lib/mySlim.h	49
src/lib/system.h	49
src/lib/system_imp.cpp	50
src/lib/system_imp.h	51
test/functional/functional_tests.cpp	56
test/functional/functional_tests.h	59
test/functional/main.cpp	54
test/unit/main.cpp	55
test/unit/mem_usage.cpp	61
test/unit/mem_usage.h	62
test/unit/unit_flow.cpp	64
test/unit/unit_flow.h	67
test/unit/unit_model.cpp	70
test/unit/unit_model.h	73
test/unit/unit_system.cpp	76
test/unit/unit_system.h	79
test/unit/unit_tests.cpp	82
test/unit/unit_tests.h	82



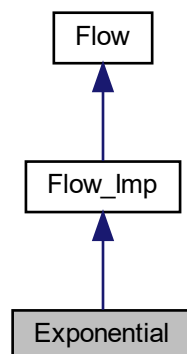
## Chapter 4

# Class Documentation

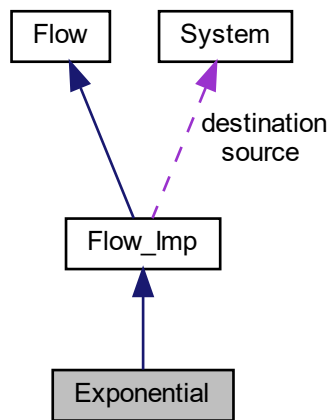
### 4.1 Exponential Class Reference

```
#include <flow_imp.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 Detailed Description

Definition at line 73 of file [flow\\_imp.h](#).

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 Exponential() [1/2]

```
Exponential::Exponential ( ) [inline]
```

Builder to create a new exponential flow.

Definition at line 79 of file [flow\\_imp.h](#).



#### 4.1.2.2 Exponential() [2/2]

```
Exponential::Exponential (
    System * source,
    System * destination ) [inline]
```

Definition at line 80 of file [flow\\_imp.h](#).

### 4.1.3 Member Function Documentation

#### 4.1.3.1 run()

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

Function to run the stream.

##### Returns

Returns double resulting from calculation performed.

Implements [Flow\\_imp](#).

Definition at line 87 of file [flow\\_imp.h](#).

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

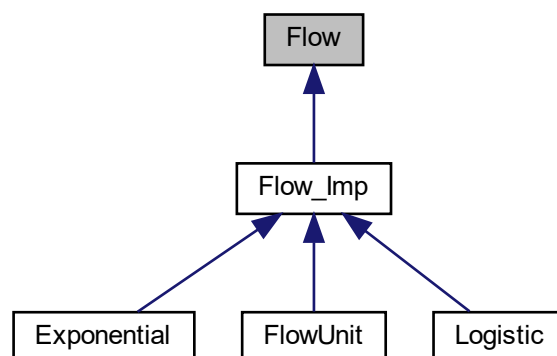
- [src/lib/flow\\_imp.h](#)

## 4.2 Flow Class Reference

File responsible for project flows.

```
#include <flow.h>
```

Inheritance diagram for Flow:



## Public Member Functions

- virtual `~Flow()`  
*Destructor to destroy the flow.*
- virtual void `setSources (System *)=0`  
*Add an input system to the stream.*
- virtual void `setDestination (System *)=0`  
*Add an exit system to the stream.*
- virtual `System * getSource ()=0`  
*Function to return an input system.*
- virtual `System * getDestination ()=0`  
*Function to return an output system.*
- virtual double `run ()=0`  
*Virtual function to run the stream.*
- virtual `Flow * operator= (Flow *)=0`  
*Function to overload operator =.*

### 4.2.1 Detailed Description

File responsible for project flows.

#### Author

Gabriel Niquini 19.1.4113

Definition at line 12 of file [flow.h](#).

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 ~Flow()

```
virtual Flow::~Flow ( ) [inline], [virtual]
```

Destructor to destroy the flow.

Definition at line 18 of file [flow.h](#).

### 4.2.3 Member Function Documentation

#### 4.2.3.1 getDestination()

```
virtual System * Flow::getDestination ( ) [pure virtual]
```

Function to return an output system.

##### Returns

Returns a [System](#) object.

Implemented in [Flow\\_Imp](#).

#### 4.2.3.2 getSource()

```
virtual System * Flow::getSource ( ) [pure virtual]
```

Function to return an input system.

##### Returns

Returns a [System](#) object.

Implemented in [Flow\\_Imp](#).

#### 4.2.3.3 operator=()

```
virtual Flow * Flow::operator= (
    Flow * ) [pure virtual]
```

Function to overload operator =.

##### Parameters

<a href="#">Flow</a>	<a href="#">Flow</a> pointer.
----------------------	-------------------------------

##### Returns

Returns flow.

Implemented in [Flow\\_Imp](#).

#### 4.2.3.4 run()

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

Virtual function to run the stream.

##### Returns

Returns value of 0.

Implemented in [Exponential](#), [Logistic](#), and [Flow\\_Imp](#).

#### 4.2.3.5 setDestination()

```
virtual void Flow::setDestination (
    System * ) [pure virtual]
```

Add an exit system to the stream.

##### Parameters

<i>system</i>	<a href="#">System</a> pointer.
---------------	---------------------------------

Implemented in [Flow\\_Imp](#).

#### 4.2.3.6 setSources()

```
virtual void Flow::setSources (
    System * ) [pure virtual]
```

Add an input system to the stream.

##### Parameters

<i>system</i>	<a href="#">System</a> pointer.
---------------	---------------------------------

Implemented in [Flow\\_Imp](#).

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

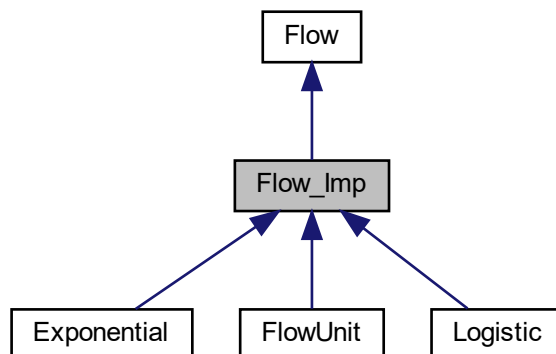
- [src/lib/flow.h](#)

## 4.3 Flow\_Imp Class Reference

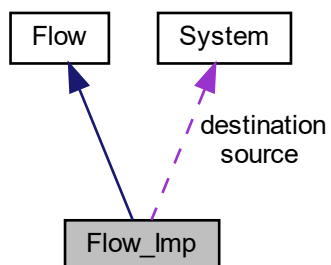
File responsible for project flows.

```
#include <flow_Imp.h>
```

Inheritance diagram for Flow\_Imp:



Collaboration diagram for Flow\_Imp:



## Public Member Functions

- [Flow\\_Imp\(\)](#)  
*Builder to create a new flow.*
- [Flow\\_Imp\(System \\*, System \\*\)](#)
- virtual [~Flow\\_Imp\(\)](#)  
*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\\_Imp](#) \* [operator=](#) ([Flow](#) \*)
- Function to overload operator =.*

## Protected Attributes

- [System](#) \* [source](#)
- [System](#) \* [destination](#)

### 4.3.1 Detailed Description

File responsible for project flows.

Author

Gabriel Niquini 19.1.4113

Definition at line 12 of file [flow\\_Imp.h](#).

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 Flow\_Imp() [1/2]

```
Flow_Imp::Flow_Imp ( )
```

Builder to create a new flow.

<Pointer of output of a system

Definition at line 3 of file [flow\\_Imp.cpp](#).

#### 4.3.2.2 Flow\_Imp() [2/2]

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

Definition at line 5 of file [flow\\_Imp.cpp](#).

#### 4.3.2.3 ~Flow\_Imp()

```
Flow_Imp::~~Flow_Imp ( ) [virtual]
```

Destructor to destroy the flow.

Definition at line 10 of file [flow\\_imp.cpp](#).

### 4.3.3 Member Function Documentation

#### 4.3.3.1 getDestination()

```
System * Flow_Imp::getDestination ( ) [virtual]
```

Function to return an output system.

##### Returns

Returns a [System](#) object.

Implements [Flow](#).

Definition at line 24 of file [flow\\_imp.cpp](#).

#### 4.3.3.2 getSource()

```
System * Flow_Imp::getSource ( ) [virtual]
```

Function to return an input system.

##### Returns

Returns a [System](#) object.

Implements [Flow](#).

Definition at line 20 of file [flow\\_imp.cpp](#).

#### 4.3.3.3 operator=()

```
Flow_Imp * Flow_Imp::operator= (
    Flow * flow ) [virtual]
```

Function to overload operator =.

**Parameters**

<a href="#">Flow_Imp</a>	Flow pointer.
--------------------------	---------------

**Returns**

Returns flow.

Implements [Flow](#).

Definition at line 28 of file [flow\\_imp.cpp](#).

**4.3.3.4 run()**

```
virtual double Flow_Imp::run ( ) [pure virtual]
```

Virtual function to run the stream.

**Returns**

Returns value of 0.

Implements [Flow](#).

Implemented in [Exponential](#), and [Logistic](#).

**4.3.3.5 setDestination()**

```
void Flow_Imp::setDestination (
    System * destination ) [virtual]
```

Add an exit system to the stream.

**Parameters**

<i>system</i>	<a href="#">System</a> pointer.
---------------	---------------------------------

Implements [Flow](#).

Definition at line 16 of file [flow\\_imp.cpp](#).



#### 4.3.3.6 setSources()

```
void Flow_Imp::setSources (
    System * source ) [virtual]
```

Add an input system to the stream.

##### Parameters

<i>system</i>	System pointer.
---------------	-----------------

Implements [Flow](#).

Definition at line 12 of file [flow\\_Imp.cpp](#).

### 4.3.4 Member Data Documentation

#### 4.3.4.1 destination

```
System* Flow_Imp::destination [protected]
```

<Pointer of entry of a system

Definition at line 15 of file [flow\\_Imp.h](#).

#### 4.3.4.2 source

```
System* Flow_Imp::source [protected]
```

Definition at line 14 of file [flow\\_Imp.h](#).

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

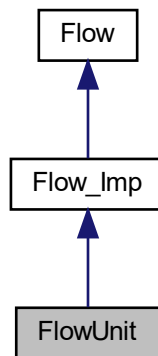
- [src/lib/flow\\_Imp.h](#)
- [src/lib/flow\\_Imp.cpp](#)

## 4.4 FlowUnit Class Reference

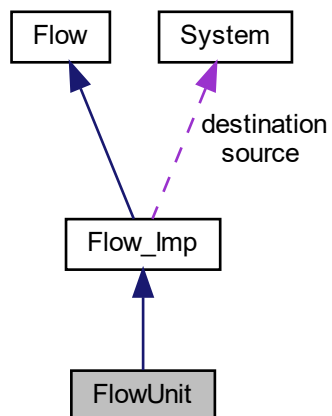
File responsible for project flows.

```
#include <flow_unit.h>
```

Inheritance diagram for FlowUnit:



Collaboration diagram for FlowUnit:



### Additional Inherited Members

#### 4.4.1 Detailed Description

File responsible for project flows.

## Author

Gabriel Niquini 19.1.4113

Definition at line 12 of file [flow\\_unit.h](#).

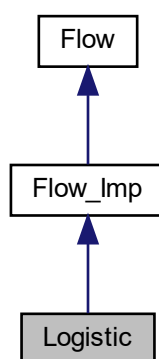
The documentation for this class was generated from the following file:

- [src/lib/flow\\_unit.h](#)

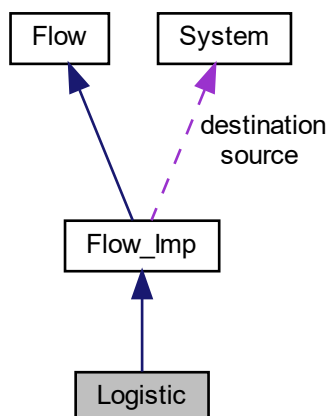
## 4.5 Logistic Class Reference

```
#include <flow_imp.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

### 4.5.1 Detailed Description

Definition at line 92 of file [flow\\_imp.h](#).

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 [Logistic\(\)](#) [1/2]

```
Logistic::Logistic ( ) [inline]
```

Builder to create a new logistical flow.

Definition at line 98 of file [flow\\_imp.h](#).

#### 4.5.2.2 [Logistic\(\)](#) [2/2]

```
Logistic::Logistic (  
    System * source,  
    System * destination ) [inline]
```

Definition at line 99 of file [flow\\_imp.h](#).

### 4.5.3 Member Function Documentation

#### 4.5.3.1 run()

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

Function to run the stream.

##### Returns

Returns double resulting from calculation performed.

Implements [Flow\\_Imp](#).

Definition at line 106 of file [flow\\_Imp.h](#).

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

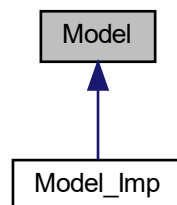
- [src/lib/flow\\_Imp.h](#)

## 4.6 Model Class Reference

File responsible for project templates.

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

Inheritance diagram for Model:



### Public Member Functions

- virtual [~Model](#) ()  
*Destructor to destroy the model.*
- virtual double [run](#) (int, int)=0  
*Function to run the model.*
- virtual [System](#) \* [createSystem](#) (string, double)=0
- virtual [Model](#) \* [operator=](#) ([Model](#) \*)=0
- virtual [System](#) \* [getSystem](#) (string name)=0  
*Function to overload operator =.*
- virtual vector< [Flow](#) \* > [getFlows](#) ()=0
- template<typename T\_FLOW >  
[Flow](#) \* [createFlow](#) ([System](#) \*source=nullptr, [System](#) \*destination=nullptr)

## Static Public Member Functions

- static [Model](#) \* [createModel](#) (string)  
*Function to create the model.*

### 4.6.1 Detailed Description

File responsible for project templates.

Author

Gabriel Niquini 19.1.4113

Definition at line 12 of file [model.h](#).

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 ~Model()

```
virtual Model::~Model ( ) [inline], [virtual]
```

Destructor to destroy the model.

Definition at line 18 of file [model.h](#).

### 4.6.3 Member Function Documentation

#### 4.6.3.1 createFlow()

```
template<typename T_FLOW >  
Flow * Model::createFlow (  
    System * source = nullptr,  
    System * destination = nullptr ) [inline]
```

Definition at line 48 of file [model.h](#).

#### 4.6.3.2 createModel()

```
Model * Model::createModel (  
    string id ) [static]
```

Function to create the model.

**Parameters**

<i>string</i>	Initial value.
---------------	----------------

**Returns**

Returns final model.

Definition at line 23 of file [model\\_imp.cpp](#).

**4.6.3.3 createSystem()**

```
virtual System * Model::createSystem (
    string ,
    double ) [pure virtual]
```

Implemented in [Model\\_imp](#).

**4.6.3.4 getFlows()**

```
virtual vector< Flow * > Model::getFlows ( ) [pure virtual]
```

Implemented in [Model\\_imp](#).

**4.6.3.5 getSystem()**

```
virtual System * Model::getSystem (
    string name ) [pure virtual]
```

Function to overload operator =.

**Parameters**

<i>model</i>	<a href="#">Model</a> pointer.
--------------	--------------------------------

**Returns**

Returns model.

Implemented in [Model\\_imp](#).

#### 4.6.3.6 operator=()

```
virtual Model * Model::operator= (
    Model * ) [pure virtual]
```

Implemented in [Model\\_Imp](#).

#### 4.6.3.7 run()

```
virtual double Model::run (
    int ,
    int ) [pure virtual]
```

Function to run the model.

##### Parameters

<i>start</i>	Initial value.
<i>finish</i>	Final value.

##### Returns

Returns final value.

Implemented in [Model\\_Imp](#).

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

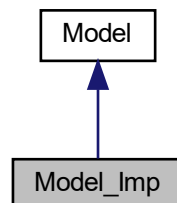
- [src/lib/model.h](#)
- [src/lib/model\\_imp.cpp](#)

## 4.7 Model\_Imp Class Reference

File responsible for project templates.

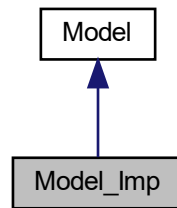
```
#include <model_imp.h>
```

Inheritance diagram for Model\_Imp:





Collaboration diagram for Model\_Imp:



## Public Member Functions

- [Model\\_Imp](#) ()  
*Builder to create a new model.*
- [Model\\_Imp](#) (string)
- virtual [~Model\\_Imp](#) ()  
*Destructor to destroy the model.*
- double [run](#) (int, int)  
*Function to run the model.*
- [System](#) \* [createSystem](#) (string, double)
- [System](#) \* [getSystem](#) (string name)  
*Function to overload operator =.*
- vector< [Flow](#) \* > [getFlows](#) ()
- [Model\\_Imp](#) \* [operator=](#) ([Model](#) \*)

## Static Public Member Functions

- static [Model](#) \* [createModel](#) (string)

## Protected Attributes

- vector< [Flow](#) \* > [flows](#)
- vector< [System](#) \* > [systems](#)

## Static Protected Attributes

- static vector< [Model](#) \* > [models](#)

### 4.7.1 Detailed Description

File responsible for project templates.

Author

Gabriel Niquini 19.1.4113

Definition at line 15 of file [model\\_Imp.h](#).

## 4.7.2 Constructor & Destructor Documentation

### 4.7.2.1 Model\_Imp() [1/2]

```
Model_Imp::Model_Imp ( )
```

Builder to create a new model.

Definition at line 7 of file [model\\_imp.cpp](#).

### 4.7.2.2 Model\_Imp() [2/2]

```
Model_Imp::Model_Imp (
    string id )
```

Definition at line 9 of file [model\\_imp.cpp](#).

### 4.7.2.3 ~Model\_Imp()

```
Model_Imp::~~Model_Imp ( ) [virtual]
```

Destructor to destroy the model.

Definition at line 13 of file [model\\_imp.cpp](#).

## 4.7.3 Member Function Documentation

### 4.7.3.1 createModel()

```
Model * Model_Imp::createModel (
    string id ) [static]
```

Definition at line 27 of file [model\\_imp.cpp](#).

#### 4.7.3.2 createSystem()

```
System * Model_Imp::createSystem (
    string name,
    double value ) [virtual]
```

Implements [Model](#).

Definition at line 56 of file [model\\_Imp.cpp](#).

#### 4.7.3.3 getFlows()

```
vector< Flow * > Model_Imp::getFlows ( ) [virtual]
```

Implements [Model](#).

Definition at line 78 of file [model\\_Imp.cpp](#).

#### 4.7.3.4 getSystem()

```
System * Model_Imp::getSystem (
    string name ) [virtual]
```

Function to overload operator =.

##### Parameters

<i>model</i>	<a href="#">Model</a> pointer.
--------------	--------------------------------

##### Returns

Returns model.

Implements [Model](#).

Definition at line 70 of file [model\\_Imp.cpp](#).

#### 4.7.3.5 operator=()

```
Model_Imp * Model_Imp::operator= (
    Model * model ) [virtual]
```

Implements [Model](#).

Definition at line 82 of file [model\\_Imp.cpp](#).

#### 4.7.3.6 run()

```
double Model_Imp::run (
    int start,
    int finish ) [virtual]
```

Function to run the model.

##### Parameters

<i>start</i>	Initial value.
<i>finish</i>	Final value.

##### Returns

Returns final value.

Implements [Model](#).

Definition at line 33 of file [model\\_imp.cpp](#).

### 4.7.4 Member Data Documentation

#### 4.7.4.1 flows

```
vector<Flow*> Model_Imp::flows [protected]
```

Definition at line 17 of file [model\\_imp.h](#).

#### 4.7.4.2 models

```
vector< Model * > Model_Imp::models [static], [protected]
```

< Systems pointer vector

Definition at line 19 of file [model\\_imp.h](#).

#### 4.7.4.3 systems

```
vector<System*> Model_Imp::systems [protected]
```

< [Flow](#) pointer vector

Definition at line 18 of file [model\\_imp.h](#).

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

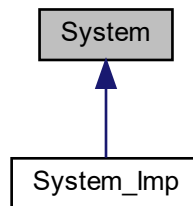
- [src/lib/model\\_imp.h](#)
- [src/lib/model\\_imp.cpp](#)

## 4.8 System Class Reference

File responsible for project systems.

```
#include <system.h>
```

Inheritance diagram for System:



### Public Member Functions

- virtual `~System()`  
*Destructor to destroy the system.*
- virtual void `setName(string)=0`  
*Add a name for the system.*
- virtual void `setValue(double)=0`  
*Add a value to the system.*
- virtual double `getValue()`=0  
*Function to return system value.*
- virtual string `getName()`=0  
*Function to return system name.*
- virtual `System * operator= (System *)`=0  
*Function to overload operator =.*

### 4.8.1 Detailed Description

File responsible for project systems.

Author

Gabriel Niquini 19.1.4113

Definition at line 15 of file [system.h](#).

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 ~System()

```
virtual System::~~System ( ) [inline], [virtual]
```

Destructor to destroy the system.

Definition at line 21 of file [system.h](#).

### 4.8.3 Member Function Documentation

#### 4.8.3.1 getName()

```
virtual string System::getName ( ) [pure virtual]
```

Function to return system name.

##### Returns

Returns a string.

Implemented in [System\\_Imp](#).

#### 4.8.3.2 getValue()

```
virtual double System::getValue ( ) [pure virtual]
```

Function to return system value.

##### Returns

Returns a double.

Implemented in [System\\_Imp](#).

#### 4.8.3.3 operator=()

```
virtual System * System::operator= (
    System * ) [pure virtual]
```

Function to overload operator =.

## Parameters

<i>flow</i>	<a href="#">System</a> pointer.
-------------	---------------------------------

## Returns

Returns a system.

Implemented in [System\\_Imp](#).

#### 4.8.3.4 setName()

```
virtual void System::setName (  
    string ) [pure virtual]
```

Add a name for the system.

## Parameters

<i>name</i>	<a href="#">System</a> name.
-------------	------------------------------

Implemented in [System\\_Imp](#).

#### 4.8.3.5 setValue()

```
virtual void System::setValue (  
    double ) [pure virtual]
```

Add a value to the system.

## Parameters

<i>value</i>	<a href="#">System</a> value.
--------------	-------------------------------

Implemented in [System\\_Imp](#).

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

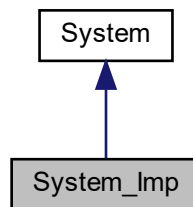
- [src/lib/system.h](#)

## 4.9 System\_Imp Class Reference

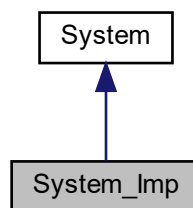
File responsible for project systems.

```
#include <system_imp.h>
```

Inheritance diagram for System\_imp:



Collaboration diagram for System\_imp:



## Public Member Functions

- [System\\_imp](#) ()  
*Builder to create a new system.*
- [System\\_imp](#) (string, double)
- virtual [~System\\_imp](#) ()  
*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\\_imp](#) \* [operator=](#) ([System](#) \*)  
*Function to overload operator =.*



## Protected Attributes

- string [name](#)
- double [value](#)

### 4.9.1 Detailed Description

File responsible for project systems.

#### Author

Gabriel Niquini 19.1.4113

Definition at line 12 of file [system\\_imp.h](#).

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 System\_Imp() [1/2]

```
System_Imp::System_Imp ( )
```

Builder to create a new system.

<Double value

Definition at line 3 of file [system\\_imp.cpp](#).

#### 4.9.2.2 System\_Imp() [2/2]

```
System_Imp::System_Imp (
    string name,
    double value )
```

Definition at line 5 of file [system\\_imp.cpp](#).

#### 4.9.2.3 ~System\_Imp()

```
System_Imp::~System_Imp ( ) [virtual]
```

Destructor to destroy the system.

Definition at line 10 of file [system\\_imp.cpp](#).

### 4.9.3 Member Function Documentation

#### 4.9.3.1 getName()

```
string System_Imp::getName ( ) [virtual]
```

Function to return system name.

##### Returns

Returns a string.

Implements [System](#).

Definition at line 20 of file [system\\_imp.cpp](#).

#### 4.9.3.2 getValue()

```
double System_Imp::getValue ( ) [virtual]
```

Function to return system value.

##### Returns

Returns a double.

Implements [System](#).

Definition at line 24 of file [system\\_imp.cpp](#).

#### 4.9.3.3 operator=()

```
System\_Imp * System_Imp::operator= (
    System * system ) [virtual]
```

Function to overload operator =.

##### Parameters

<i>flow</i>	<a href="#">System</a> pointer.
-------------	---------------------------------

#### Returns

Returns a system.

Implements [System](#).

Definition at line 28 of file [system\\_Imp.cpp](#).

#### 4.9.3.4 setName()

```
void System_Imp::setName (
    string name ) [virtual]
```

Add a name for the system.

#### Parameters

<i>name</i>	<a href="#">System</a> name.
-------------	------------------------------

Implements [System](#).

Definition at line 12 of file [system\\_Imp.cpp](#).

#### 4.9.3.5 setValue()

```
void System_Imp::setValue (
    double value ) [virtual]
```

Add a value to the system.

#### Parameters

<i>value</i>	<a href="#">System</a> value.
--------------	-------------------------------

Implements [System](#).

Definition at line 16 of file [system\\_Imp.cpp](#).

### 4.9.4 Member Data Documentation

#### 4.9.4.1 name

```
string System_Imp::name [protected]
```

Definition at line 14 of file [system\\_Imp.h](#).

#### 4.9.4.2 value

```
double System_Imp::value [protected]
```

<String name

Definition at line 15 of file [system\\_imp.h](#).

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

- [src/lib/system\\_imp.h](#)
- [src/lib/system\\_imp.cpp](#)

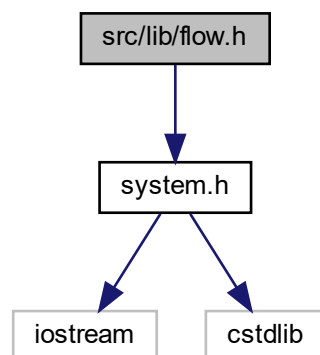
## Chapter 5

# File Documentation

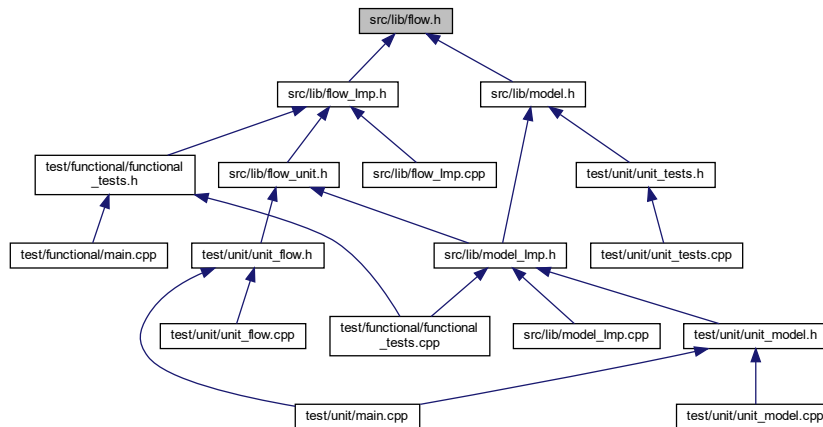
### 5.1 src/lib/flow.h File Reference

```
#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.*

## 5.2 flow.h

[Go to the documentation of this file.](#)

```

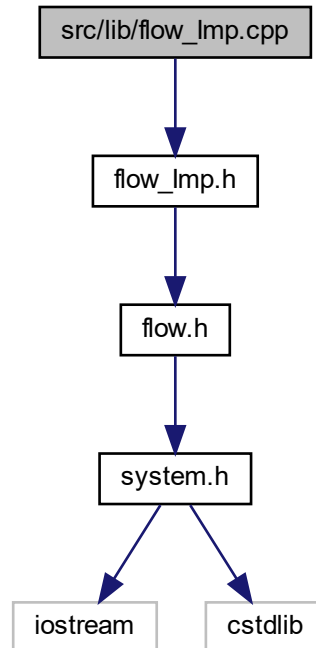
00001 #ifndef FLOW_H
00002 #define FLOW_H
00003
00004 #include "system.h"
00005
00012 class Flow{
00013     public:
00018         virtual ~Flow(){};
00019
00025         virtual void setSources(System*) = 0;
00026
00027
00032         virtual void setDestination(System*) = 0;
00033
00039         virtual System* getSource() = 0;
00040
00046         virtual System* getDestination() = 0;
00047
00053         virtual double run() = 0;
00054
00061         virtual Flow* operator=(Flow*) = 0;
00062 };
00063
00064 #endif

```

## 5.3 src/lib/flow\_Imp.cpp File Reference

```
#include "flow_Imp.h"
```

Include dependency graph for flow\_Imp.cpp:



## 5.4 flow\_Imp.cpp

[Go to the documentation of this file.](#)

```

00001 #include "flow_Imp.h"
00002
00003 Flow_Imp::Flow_Imp() {}
00004
00005 Flow_Imp::Flow_Imp(System* source, System* destination){
00006     this->source = source;
00007     this->destination = destination;
00008 }
00009
00010 Flow_Imp::~Flow_Imp() {}
00011
00012 void Flow_Imp::setSources(System* source){
00013     this->source = source;
00014 }
00015
00016 void Flow_Imp::setDestination(System* destination){
00017     this->destination = destination;
00018 }
00019
00020 System* Flow_Imp::getSource(){
00021     return this->source;
00022 }
00023
00024 System* Flow_Imp::getDestination(){
00025     return this->destination;
00026 }
00027

```

```

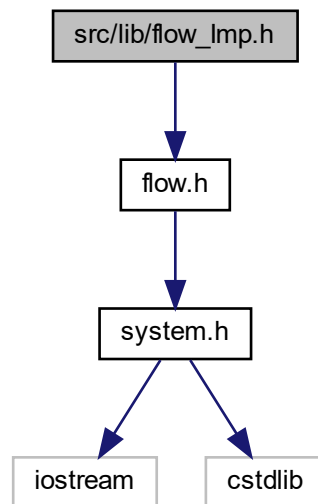
00028 Flow_Imp* Flow_Imp::operator=(Flow* flow) {
00029     if (this == flow)
00030         return this;
00031
00032     this->source = flow->getSource();
00033     this->destination = flow->getSource();
00034     return this;
00035 }

```

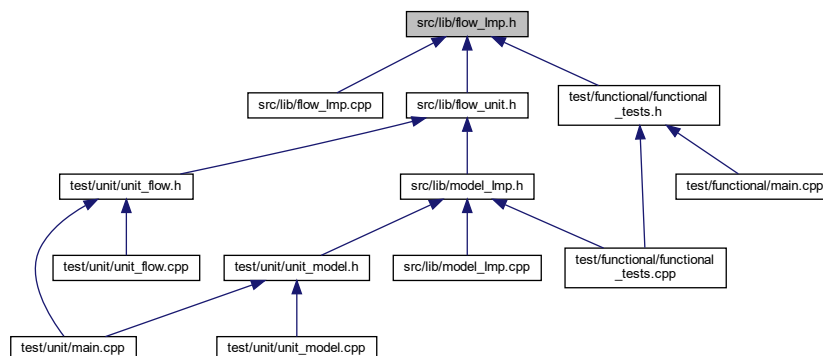
## 5.5 src/lib/flow\_Imp.h File Reference

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

Include dependency graph for flow\_Imp.h:



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





## Classes

- class [Flow\\_Imp](#)  
*File responsible for project flows.*
- class [Exponential](#)
- class [Logistic](#)

## 5.6 flow\_Imp.h

[Go to the documentation of this file.](#)

```

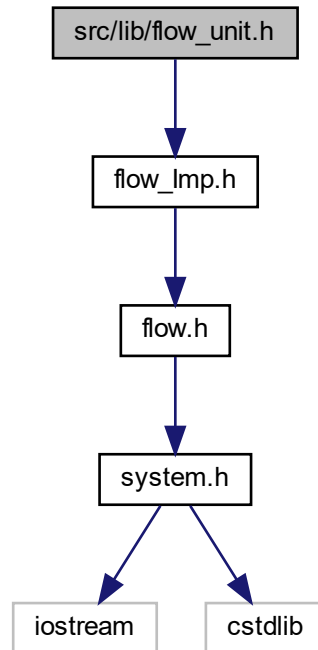
00001 #ifndef FLOW_IMP_H
00002 #define FLOW_IMP_H
00003
00004 #include "flow.h"
00005
00012 class Flow_Imp : public Flow{
00013     protected:
00014         System* source;
00015         System* destination;
00016     public:
00021         Flow_Imp();
00022         Flow_Imp(System*, System*);
00023
00028         virtual ~Flow_Imp();
00029
00035         void setSources(System*);
00036
00041         void setDestination(System*);
00042
00048         System* getSource();
00049
00055         System* getDestination();
00056
00062         virtual double run() = 0;
00063
00070         Flow_Imp* operator=(Flow*);
00071 };
00072
00073 class Exponential: public Flow_Imp{
00074     public:
00079         Exponential() {};
00080         Exponential(System* source, System* destination):Flow_Imp(source,destination){};
00081
00087         double run(){
00088             return getSource()->getValue()*0.01;
00089         }
00090 };
00091
00092 class Logistic: public Flow_Imp{
00093     public:
00098         Logistic() {};
00099         Logistic(System* source, System* destination):Flow_Imp(source,destination){};
00100
00106         double run(){
00107             return getDestination()->getValue()*0.01*(1-(getDestination()->getValue())/70);
00108         }
00109 };
00110
00111 #endif

```

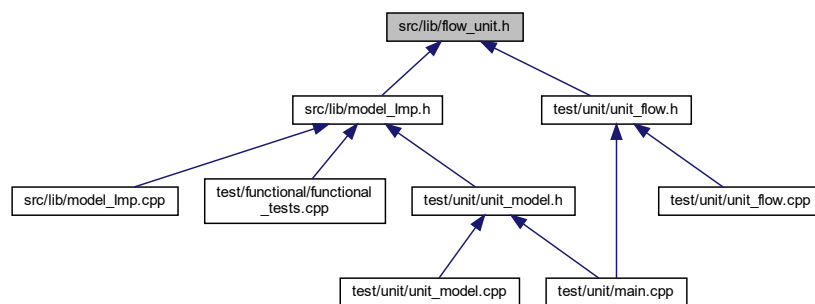
## 5.7 src/lib/flow\_unit.h File Reference

```
#include "flow_imp.h"
```

Include dependency graph for flow\_unit.h:



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



## Classes

- class [FlowUnit](#)

*File responsible for project flows.*

## 5.8 flow\_unit.h

[Go to the documentation of this file.](#)

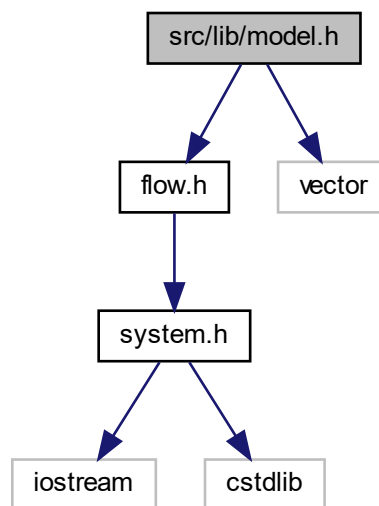
```
00001 #ifndef FLOW_UNIT_H
00002 #define FLOW_UNIT_H
00003
00004 #include "flow_imp.h"
00005
00012 class FlowUnit : public Flow_Imp{
00018     double run(){
00019         return 0;
00020     }
00021 };
00022
00023 #endif
```

## 5.9 src/lib/model.h File Reference

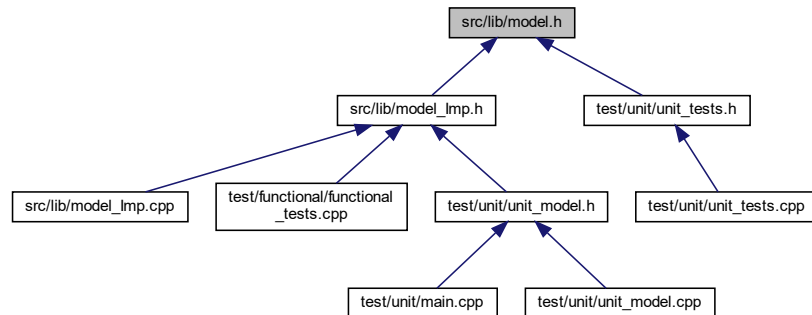
```
#include "flow.h"
```

```
#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.*

## 5.10 model.h

[Go to the documentation of this file.](#)

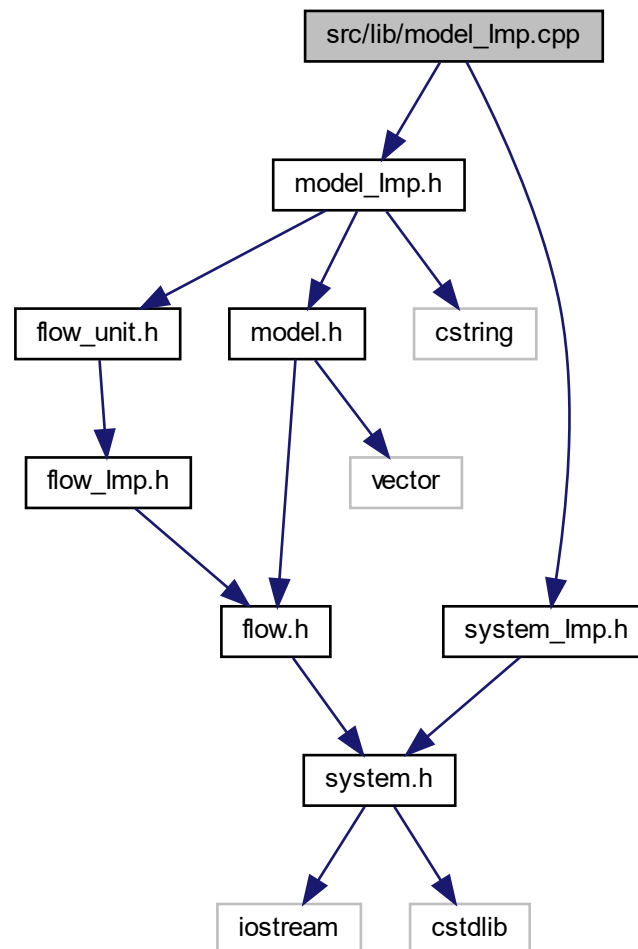
```

00001 #ifndef MODEL_H
00002 #define MODEL_H
00003 #include "flow.h"
00004 #include <vector>
00005
00012 class Model{
00013     public:
00018         virtual ~Model(){};
00019
00027         virtual double run(int,int) = 0;
00028
00035         static Model* createModel(string);
00036         virtual System* createSystem(string,double) = 0;
00037         virtual Model* operator=(Model*) = 0;
00038
00045         virtual System* getSystem(string name) = 0;
00046         virtual vector<Flow*> getFlows() = 0;
00047         template <typename T_FLOW>
00048         Flow* createFlow(System* source = nullptr, System* destination = nullptr){
00049             Flow* flow = new T_FLOW();
00050             flow->setSources(source);
00051             flow->setDestination(destination);
00052             add(flow);
00053             return flow;
00054         }
00055
00056     private:
00061         virtual void add(System*) = 0;
00062
00067         virtual void add(Flow*) = 0;
00068 };
00069
00070 #endif

```

## 5.11 src/lib/model\_Imp.cpp File Reference

```
#include "model_Imp.h"
#include "system_Imp.h"
Include dependency graph for model_Imp.cpp:
```



## 5.12 model\_Imp.cpp

[Go to the documentation of this file.](#)

```
00001 #include "model_Imp.h"
00002 #include "system_Imp.h"
00003
00004 //Global Variable
00005 vector<Model*> Model_Imp:: models;
00006
00007 Model_Imp::Model_Imp() {}
00008
00009 Model_Imp::Model_Imp(string id) {
00010     this->id = id;
00011 }
```

```

00012
00013 Model_Imp::~Model_Imp(){
00014     for (auto it = flows.begin(); it != flows.end(); it++){
00015         delete *it;
00016     for (auto it = systems.begin(); it != systems.end(); it++){
00017         delete *it;
00018     this->flows.clear();
00019     this->systems.clear();
00020
00021 }
00022
00023 Model* Model::createModel(string id){
00024     return Model_Imp::createModel(id);
00025 }
00026
00027 Model* Model_Imp::createModel(string id){
00028     Model* m = new Model_Imp(id);
00029     models.push_back(m);
00030     return m;
00031 }
00032
00033 double Model_Imp::run(int start,int finish){
00034     vector<double> values;
00035     System* source;
00036     System* destination;
00037
00038     int size = flows.size();
00039
00040     for (int i = start; i < finish; i++){
00041         for(int j = 0; j < size; j++){
00042             values.push_back(flows[j]->run());
00043         }
00044         for(int k = 0; k < size; k++){
00045             source = flows[k]->getSource();
00046             source->setValue(source->getValue() - values[k]);
00047             destination = flows[k]->getDestination();
00048             destination->setValue(destination->getValue() + values[k]);
00049         }
00050         values.clear();
00051     }
00052
00053     return values[finish];
00054 }
00055
00056 System* Model_Imp::createSystem(string name, double value){
00057     System* s = new System_Imp(name,value);
00058     this->add(s);
00059     return s;
00060 }
00061
00062 void Model_Imp::add(System* system){
00063     this->systems.push_back(system);
00064 }
00065
00066 void Model_Imp::add(Flow* flow){
00067     this->flows.push_back(flow);
00068 }
00069
00070 System* Model_Imp::getSystem(string name){
00071     for (vector<System*>::iterator it= systems.begin(); it != systems.end(); it++){
00072         if(name == (*it)->getName())
00073             return *it;
00074     }
00075     return NULL;
00076 }
00077
00078 vector<Flow*> Model_Imp::getFlows(){
00079     return this->flows;
00080 }
00081
00082 Model_Imp* Model_Imp::operator=(Model* model){
00083     if(this == model)
00084         return this;
00085
00086     for (vector<System*>::iterator it= systems.begin(); it != systems.end(); it++){
00087         delete *it;
00088     }
00089     this->systems.clear();
00090
00091     for (vector<Flow*>::iterator it= flows.begin(); it != flows.end(); it++){
00092         delete *it;
00093     }
00094     this->flows.clear();
00095
00096     return this;
00097
00098

```

```
00099 }
```

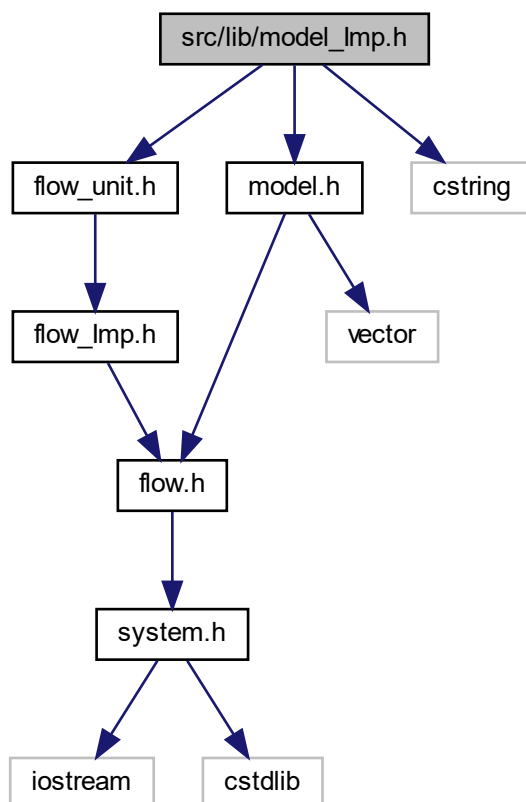
## 5.13 src/lib/model\_imp.h File Reference

```
#include "flow_unit.h"
```

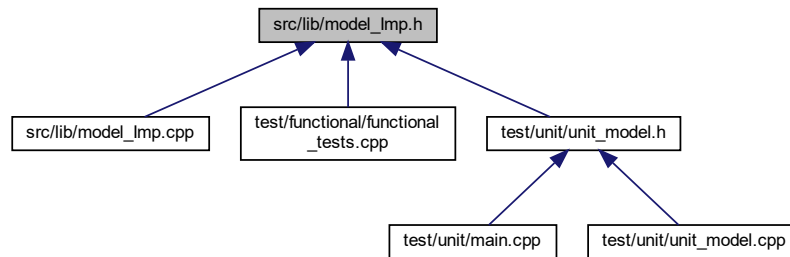
```
#include "model.h"
```

```
#include <cstring>
```

Include dependency graph for model\_imp.h:



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



## Classes

- class [Model\\_Imp](#)

*File responsible for project templates.*

## 5.14 model\_imp.h

[Go to the documentation of this file.](#)

```

00001 #ifndef MODEL_IMP_H
00002 #define MODEL_IMP_H
00003
00004 #include "flow_unit.h"
00005 #include "model.h"
00006 #include <cstring>
00007
00008
00015 class Model_Imp : public Model{
00016     protected:
00017         vector<Flow*> flows;
00018         vector<System*> systems;
00019         static vector<Model*> models;
00020     public:
00025         Model_Imp();
00026
00027         Model_Imp(string);
00028
00033         virtual ~Model_Imp();
00034
00035         static Model* createModel(string);
00036
00044         double run(int,int);
00045
00046         System* createSystem(string,double);
00047
00054         System* getSystem(string name);
00055         vector<Flow*> getFlows();
00056         Model_Imp* operator=(Model*);
00057
00058     private:
00063         void add(System*);
00064
00069         void add(Flow*);
00070         string id;
00071 };
00072
00073 #endif

```





## Classes

- class [System](#)

*File responsible for project systems.*

## 5.20 system.h

[Go to the documentation of this file.](#)

```

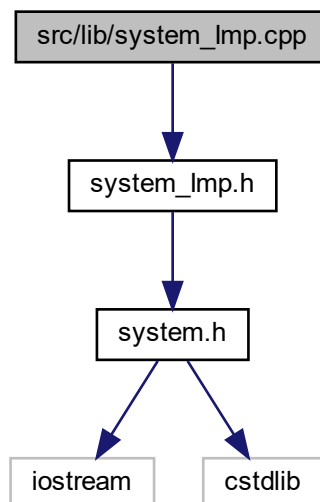
00001 #ifndef SYSTEM_H
00002 #define SYSTEM_H
00003
00004 #include <iostream>
00005 #include <cstdlib>
00006
00007 using namespace std;
00008
00015 class System{
00016     public:
00021         virtual ~System(){};
00022
00027         virtual void setName(string) = 0;
00028
00033         virtual void setValue(double) = 0;
00034
00040         virtual double getValue() = 0;
00041
00047         virtual string getName() = 0;
00048
00055         virtual System* operator=(System*) = 0;
00056 };
00057
00058 #endif

```

## 5.21 src/lib/system\_imp.cpp File Reference

```
#include "system_imp.h"
```

Include dependency graph for system\_imp.cpp:



## 5.22 system\_Imp.cpp

[Go to the documentation of this file.](#)

```

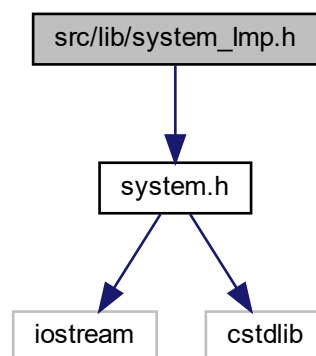
00001 #include "system_Imp.h"
00002
00003 System_Imp::System_Imp() {}
00004
00005 System_Imp::System_Imp(string name, double value){
00006     this->name = name;
00007     this->value = value;
00008 }
00009
00010 System_Imp::~System_Imp() {}
00011
00012 void System_Imp::setName(string name){
00013     this->name = name;
00014 }
00015
00016 void System_Imp::setValue(double value){
00017     this->value = value;
00018 }
00019
00020 string System_Imp::getName(){
00021     return this->name;
00022 }
00023
00024 double System_Imp::getValue(){
00025     return this->value;
00026 }
00027
00028 System_Imp* System_Imp::operator=(System* system){
00029     if (this == system)
00030         return this;
00031     this->name = system->getName();
00032     this->value = system->getValue();
00033     return this;
00034 }

```

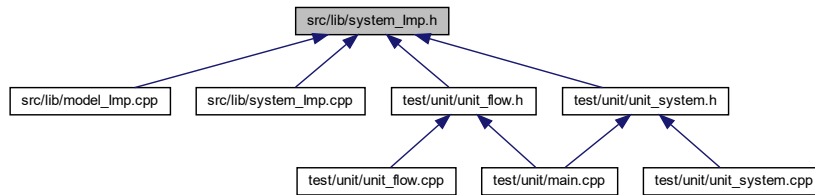
## 5.23 src/lib/system\_Imp.h File Reference

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

Include dependency graph for system\_Imp.h:



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



## Classes

- class [System\\_Imp](#)

*File responsible for project systems.*

## 5.24 system\_imp.h

[Go to the documentation of this file.](#)

```

00001 #ifndef SYSTEM_IMP_H
00002 #define SYSTEM_IMP_H
00003
00004 #include "system.h"
00005
00012 class System_Imp : public System{
00013     protected:
00014         string name;
00015         double value;
00016     public:
00021         System_Imp();
00022         System_Imp(string,double);
00023
00028         virtual ~System_Imp();
00029
00034         void setName(string);
00035
00040         void setValue(double);
00041
00047         double getValue();
00048
00054         string getName();
00055
00062         System_Imp* operator=(System*);
00063 };
00064
00065 #endif

```

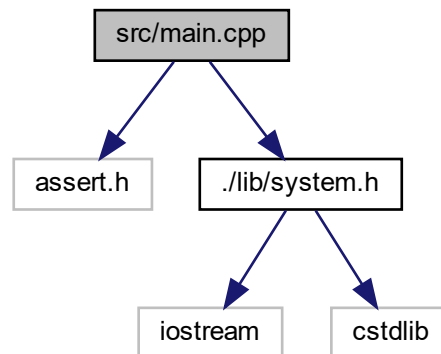
## 5.25 src/main.cpp File Reference

```

#include <assert.h>
#include "../lib/system.h"

```

Include dependency graph for main.cpp:



## Functions

- int `main` ()

### 5.25.1 Function Documentation

#### 5.25.1.1 main()

```
int main ( )
```

Definition at line 4 of file [main.cpp](#).

## 5.26 main.cpp

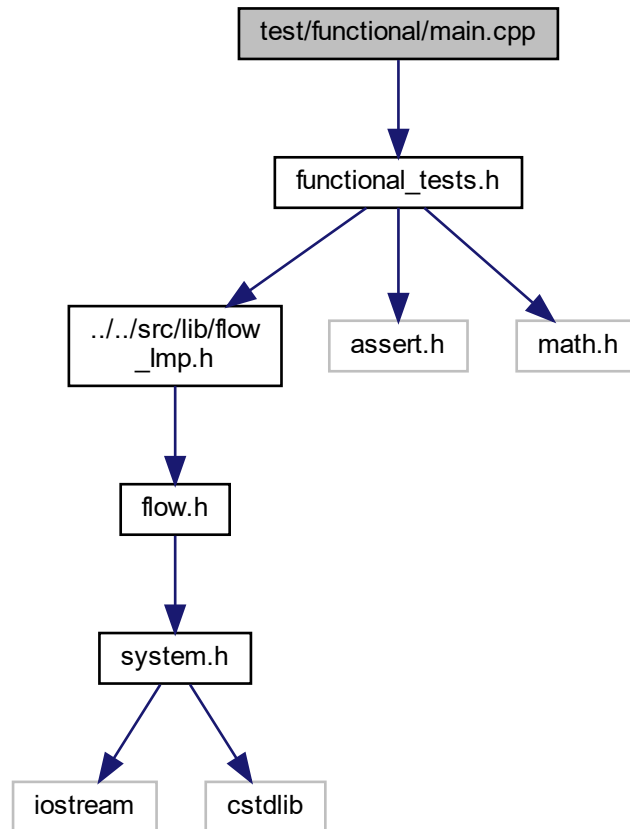
[Go to the documentation of this file.](#)

```
00001 #include <assert.h>
00002 #include "../lib/system.h"
00003
00004 int main () { return 0; }
```

## 5.27 test/functional/main.cpp File Reference

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

Include dependency graph for main.cpp:



### Functions

- int [main](#) ()

#### 5.27.1 Function Documentation

##### 5.27.1.1 main()

```
int main ( )
```

Definition at line 3 of file [main.cpp](#).

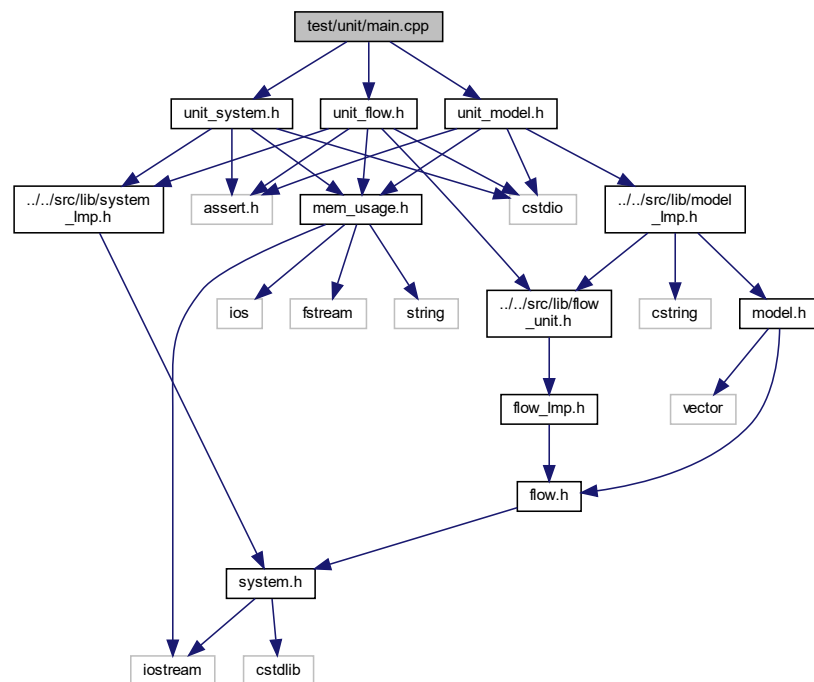
## 5.28 main.cpp

[Go to the documentation of this file.](#)

```
00001 #include "functional_tests.h"
00002
00003 int main () {
00004
00005     exponentialFunctionalTest();
00006     logisticalFunctionalTest();
00007     complexFunctionalTest();
00008
00009     return 0;
00010 }
```

## 5.29 test/unit/main.cpp File Reference

```
#include "unit_system.h"
#include "unit_flow.h"
#include "unit_model.h"
Include dependency graph for main.cpp:
```



### Functions

- int [main](#) ()

### 5.29.1 Function Documentation

### 5.29.1.1 main()

```
int main ( )
```

Definition at line 5 of file [main.cpp](#).

## 5.30 main.cpp

[Go to the documentation of this file.](#)

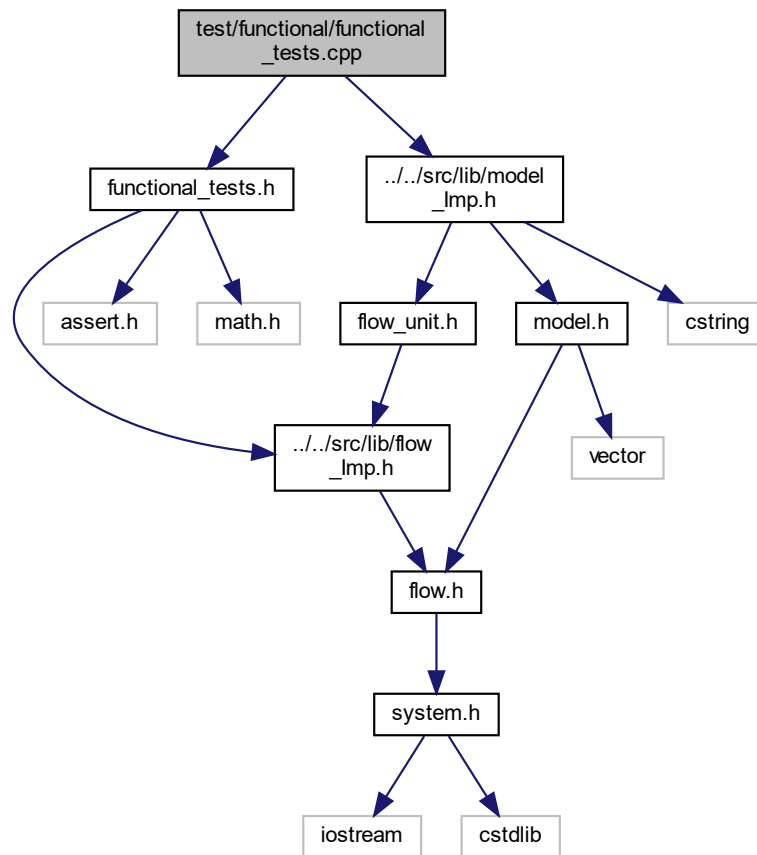
```
00001 #include "unit_system.h"
00002 #include "unit_flow.h"
00003 #include "unit_model.h"
00004
00005 int main () {
00006
00007     run_unit_test_System();
00008     run_unit_test_Flow();
00009     run_unit_test_Model();
00010
00011     return 0;
00012 }
```

## 5.31 test/functional/functional\_tests.cpp File Reference

```
#include "functional_tests.h"
#include "../src/lib/model_Imp.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.*

### 5.31.1 Function Documentation

#### 5.31.1.1 `complexFuncionalTest()`

```
void complexFuncionalTest ( )
```

Complex functional test.

Definition at line 29 of file `functional_tests.cpp`.

### 5.31.1.2 exponentialFuncionalTest()

```
void exponentialFuncionalTest ( )
```

File responsible for functional testing.

Author

Gabriel Niquini 19.1.4113

[Exponential](#) functional test.

Definition at line 4 of file [functional\\_tests.cpp](#).

### 5.31.1.3 logisticalFuncionalTest()

```
void logisticalFuncionalTest ( )
```

Logistics functional test.

Definition at line 16 of file [functional\\_tests.cpp](#).

## 5.32 functional\_tests.cpp

[Go to the documentation of this file.](#)

```
00001 #include "functional_tests.h"
00002 #include "../src/lib/model_imp.h"
00003
00004 void exponentialFuncionalTest() {
00005     Model* Modelexponential = Model::createModel("Model pops");
00006     System* pop1 = Modelexponential->createSystem("pop1", 100.0);
00007     System* pop2 = Modelexponential->createSystem("pop2", 0.0);
00008     Flow* f = Modelexponential->createFlow<Exponential>(pop1, pop2);
00009     Modelexponential->run(0, 100);
00010     assert(abs(pop1->getValue() - 36.6032) < 0.0001);
00011     assert(abs(pop2->getValue() - 63.3968) < 0.0001);
00012
00013     cout << endl << "Exponential test OK!" << endl;
00014 }
00015
00016 void logisticalFuncionalTest() {
00017     Model* ModelLogistic = Model::createModel("Model Logistic");
00018     System* p1 = ModelLogistic->createSystem("p1", 100.0);
00019     System* p2 = ModelLogistic->createSystem("p2", 10.0);
00020     Flow* l = ModelLogistic->createFlow<Logistic>(p1, p2);
00021     ModelLogistic->run(0, 100);
00022
00023     assert(abs(p1->getValue() - 88.2167) < 0.0001);
00024     assert(abs(p2->getValue() - 21.7834) < 0.0001);
00025
00026     cout << endl << "Logistic test OK!" << endl;
00027 }
00028
00029 void complexFuncionalTest() {
00030     Model* model = Model::createModel("Model Complex");
00031     System* q1 = model->createSystem("q1", 100.0);
00032     System* q2 = model->createSystem("q2", 0.0);
00033     System* q3 = model->createSystem("q3", 100.0);
00034     System* q4 = model->createSystem("q4", 0.0);
00035     System* q5 = model->createSystem("q5", 0.0);
00036
00037     Flow* f = model->createFlow<Exponential>(q1, q2);
00038     Flow* g = model->createFlow<Exponential>(q1, q3);
00039     Flow* r = model->createFlow<Exponential>(q2, q5);
```

```

00040     Flow* t = model->createFlow<Exponential>(q2,q3);
00041     Flow* u = model->createFlow<Exponential>(q3,q4);
00042     Flow* v = model->createFlow<Exponential>(q4,q1);
00043
00044     model->run(0,100);
00045
00046     assert(abs((q1->getValue() - 31.8513)) < 0.0001);
00047     assert(abs((q2->getValue() - 18.4003)) < 0.0001);
00048     assert(abs((q3->getValue() - 77.1143)) < 0.0001);
00049     assert(abs((q4->getValue() - 56.1728)) < 0.0001);
00050     assert(abs((q5->getValue() - 16.4612)) < 0.0001);
00051
00052     cout << endl << "Complex test OK!" << endl;
00053 }

```

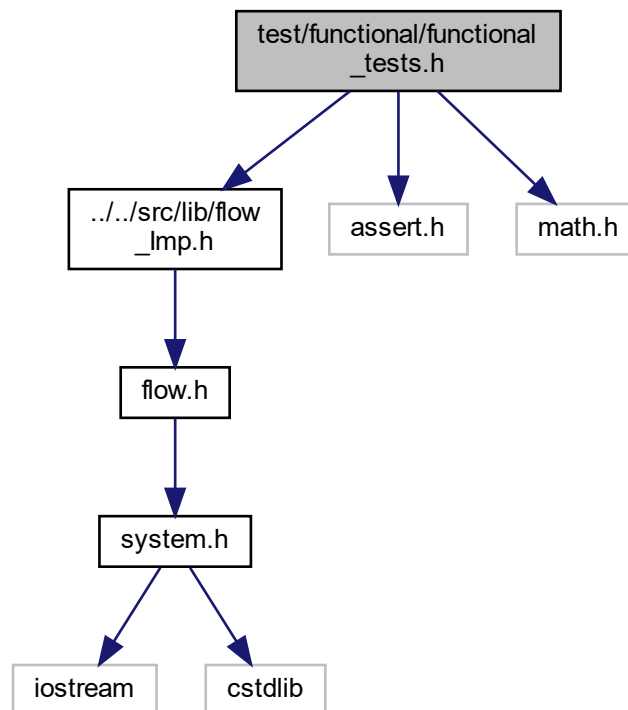
## 5.33 test/functional/functional\_tests.h File Reference

```

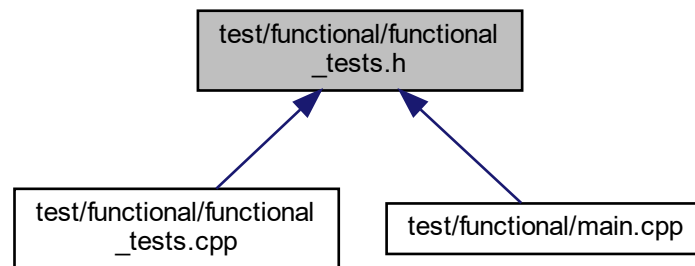
#include "../src/lib/flow_Imp.h"
#include <assert.h>
#include <math.h>

```

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.*

### 5.33.1 Function Documentation

#### 5.33.1.1 [complexFuncionalTest](#)()

```
void complexFuncionalTest ( )
```

Complex functional test.

Definition at line [29](#) of file [functional\\_tests.cpp](#).

#### 5.33.1.2 [exponentialFuncionalTest](#)()

```
void exponentialFuncionalTest ( )
```

File responsible for functional testing.

**Author**

Gabriel Niquini 19.1.4113

[Exponential](#) functional test.

Definition at line [4](#) of file [functional\\_tests.cpp](#).

## 5.33.1.3 logisticalFuncionalTest()

```
void logisticalFuncionalTest ( )
```

Logistics functional test.

Definition at line 16 of file [functional\\_tests.cpp](#).

## 5.34 functional\_tests.h

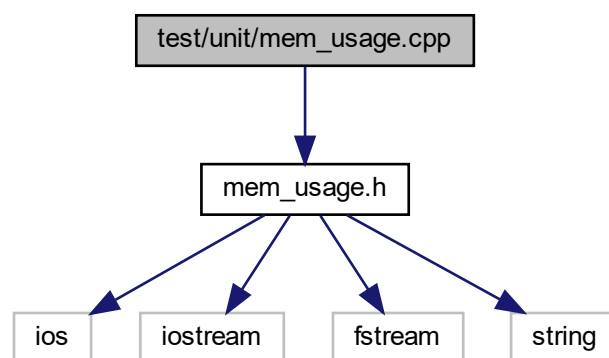
[Go to the documentation of this file.](#)

```
00001 #ifndef FUNCTIONAL_TESTS_H
00002 #define FUNCTIONAL_TESTS_H
00003 #include "../src/lib/flow_imp.h"
00004
00005 #include <assert.h>
00006 #include <math.h>
00007
00008
00019 void exponentialFuncionalTest ();
00020
00025 void logisticalFuncionalTest ();
00026
00031 void complexFuncionalTest ();
00032
00033 #endif
```

## 5.35 test/unit/mem\_usage.cpp File Reference

```
#include "mem_usage.h"
```

Include dependency graph for mem\_usage.cpp:



## Functions

- void [mem\\_usage](#) (double &vm\_usage, double &resident\_set)

### 5.35.1 Function Documentation

#### 5.35.1.1 mem\_usage()

```
void mem_usage (
    double & vm_usage,
    double & resident_set )
```

Definition at line 3 of file [mem\\_usage.cpp](#).

## 5.36 mem\_usage.cpp

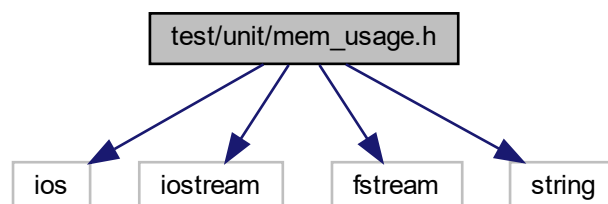
[Go to the documentation of this file.](#)

```
00001 #include "mem_usage.h"
00002
00003 void mem_usage(double& vm_usage, double& resident_set) {
00004     vm_usage = 0.0;
00005     resident_set = 0.0;
00006     ifstream stat_stream("/proc/self/stat",ios_base::in); //get info from proc directory
00007
00008     //create some variables to get info
00009     string pid, comm, state, ppid, pgrp, session, tty_nr;
00010     string tpgid, flags, minflt, cminflt, majflt, cmajflt;
00011     string utime, stime, cutime, cstime, priority, nice;
00012     string O, itrealvalue, starttime;
00013     unsigned long vsize;
00014     long rss;
00015
00016     stat_stream >> pid >> comm >> state >> ppid >> pgrp >> session >> tty_nr
00017     >> tpgid >> flags >> minflt >> cminflt >> majflt >> cmajflt
00018     >> utime >> stime >> cutime >> cstime >> priority >> nice
00019     >> O >> itrealvalue >> starttime >> vsize >> rss; // don't care about the rest
00020
00021     stat_stream.close();
00022     vm_usage = vsize / 1024.0;
00023 }
```

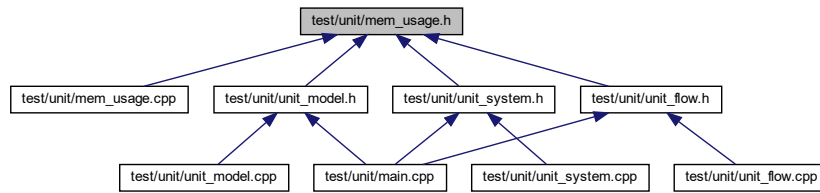
## 5.37 test/unit/mem\_usage.h File Reference

```
#include <ios>
#include <iostream>
#include <fstream>
#include <string>
```

Include dependency graph for mem\_usage.h:



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



## Functions

- void [mem\\_usage](#) (double &vm\_usage, double &resident\_set)

### 5.37.1 Function Documentation

#### 5.37.1.1 mem\_usage()

```

void mem_usage (
    double & vm_usage,
    double & resident_set )
  
```

Definition at line 3 of file [mem\\_usage.cpp](#).

## 5.38 mem\_usage.h

[Go to the documentation of this file.](#)

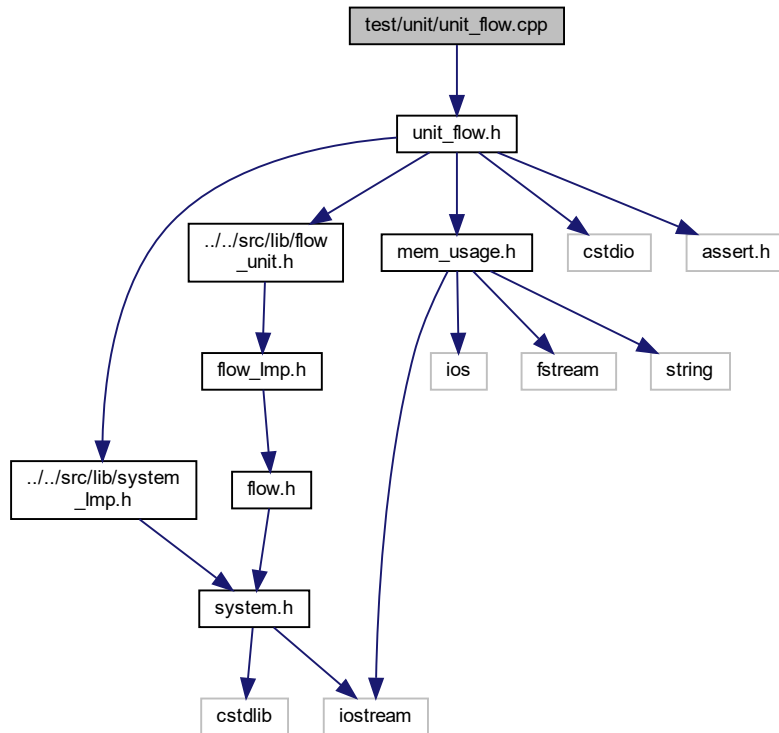
```

00001 #include <ios>
00002 #include <iostream>
00003 #include <fstream>
00004 #include <string>
00005
00006 using namespace std;
00007
00008 void mem_usage(double& vm_usage, double& resident_set);
  
```

## 5.39 test/unit/unit\_flow.cpp File Reference

```
#include "unit_flow.h"
```

Include dependency graph for unit\_flow.cpp:



### Functions

- void [unit\\_Flow\\_constructor](#) (void)
- void [unit\\_Flow\\_destructor](#) (void)
- void [unit\\_Flow\\_setSource](#) (void)
- void [unit\\_Flow\\_setDestination](#) (void)
- void [unit\\_Flow\\_getSource](#) (void)
- void [unit\\_Flow\\_getDestination](#) (void)
- void [unit\\_Flow\\_operator](#) (void)
- void [run\\_unit\\_test\\_Flow](#) (void)

### 5.39.1 Function Documentation



#### 5.39.1.1 run\_unit\_test\_Flow()

```
void run_unit_test_Flow (  
    void )
```

Definition at line 64 of file [unit\\_flow.cpp](#).

#### 5.39.1.2 unit\_Flow\_constructor()

```
void unit_Flow_constructor (  
    void )
```

Definition at line 3 of file [unit\\_flow.cpp](#).

#### 5.39.1.3 unit\_Flow\_destructor()

```
void unit_Flow_destructor (  
    void )
```

Definition at line 10 of file [unit\\_flow.cpp](#).

#### 5.39.1.4 unit\_Flow\_getDestination()

```
void unit_Flow_getDestination (  
    void )
```

Definition at line 49 of file [unit\\_flow.cpp](#).

#### 5.39.1.5 unit\_Flow\_getSource()

```
void unit_Flow_getSource (  
    void )
```

Definition at line 42 of file [unit\\_flow.cpp](#).

#### 5.39.1.6 unit\_Flow\_operator()

```
void unit_Flow_operator (  
    void )
```

Definition at line 56 of file [unit\\_flow.cpp](#).

### 5.39.1.7 unit\_Flow\_setDestination()

```
void unit_Flow_setDestination (
    void )
```

Definition at line 33 of file [unit\\_flow.cpp](#).

### 5.39.1.8 unit\_Flow\_setSource()

```
void unit_Flow_setSource (
    void )
```

Definition at line 24 of file [unit\\_flow.cpp](#).

## 5.40 unit\_flow.cpp

[Go to the documentation of this file.](#)

```
00001 #include "unit_flow.h"
00002
00003 void unit_Flow_constructor(void){
00004     System*s = new System_Imp("n1",40);
00005     System*t = new System_Imp("n2",6);
00006     Flow* f = new Logistic(s,t);
00007     assert(f->getDestination()==t);
00008 }
00009
00010 void unit_Flow_destructor(void){
00011     double memoryBefore, memoryAfter, rss;
00012
00013     mem_usage(memoryBefore, rss);
00014
00015     Flow* f = new FlowUnit();
00016     delete f;
00017
00018     mem_usage(memoryAfter, rss);
00019
00020     assert(memoryBefore == memoryAfter);
00021     cout << "Verification: destructor of Flow. OK!" << endl;
00022 }
00023
00024 void unit_Flow_setSource(void){
00025     Flow* f = new FlowUnit();
00026     System* destination = new System_Imp();
00027     System* source = new System_Imp();
00028     f->setDestination(destination);
00029     f->setSources(source);
00030     assert(f->getSource() == source);
00031 }
00032
00033 void unit_Flow_setDestination(void){
00034     Flow* f = new FlowUnit();
00035     System* destination = new System_Imp();
00036     System* source = new System_Imp();
00037     f->setDestination(destination);
00038     f->setSources(source);
00039     assert(f->getDestination() == destination);
00040 }
00041
00042 void unit_Flow_getSource(void){
00043     System* s1 = new System_Imp();
00044     System* s2 = new System_Imp();
00045     Flow* f = new Exponential(s1,s2);
00046     assert(f->getSource() == s1);
00047 }
00048
00049 void unit_Flow_getDestination(void){
00050     System* s1 = new System_Imp();
00051     System* s2 = new System_Imp();
00052     Flow* f = new Exponential(s1,s2);
```

```

00053     assert(f->getDestination() == s2);
00054 }
00055
00056 void unit_Flow_operator(void){
00057     System* s1 = new System_Imp();
00058     System* s2 = new System_Imp();
00059     Flow* f = new Exponential(s1,s2);
00060     Flow* test = f;
00061     assert(f->getDestination() == test->getDestination());
00062 }
00063
00064 void run_unit_test_Flow(void){
00065     unit_Flow_constructor();
00066     unit_Flow_destructor();
00067     unit_Flow_setSource();
00068     unit_Flow_setDestination();
00069     unit_Flow_getSource();
00070     unit_Flow_getDestination();
00071     unit_Flow_operator();
00072 }

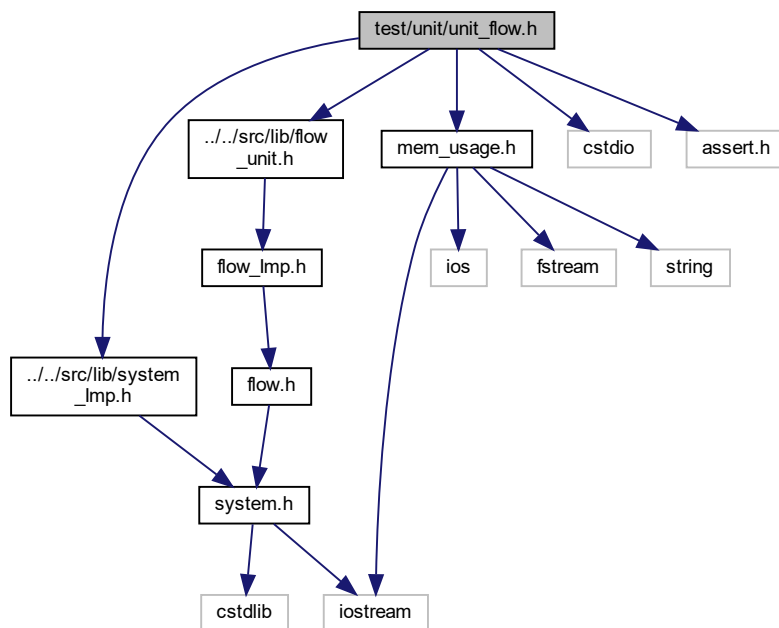
```

## 5.41 test/unit/unit\_flow.h File Reference

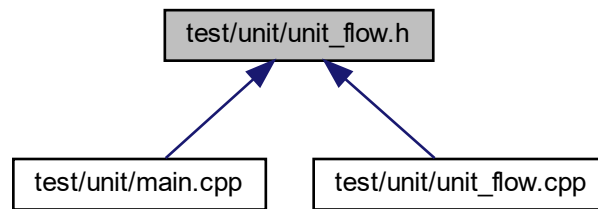
```

#include "../src/lib/system_Imp.h"
#include "../src/lib/flow_unit.h"
#include "mem_usage.h"
#include <cstdio>
#include <assert.h>
Include dependency graph for unit_flow.h:

```



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



## Functions

- void [unit\\_Flow\\_constructor](#) (void)
- void [unit\\_Flow\\_destructor](#) (void)
- void [unit\\_Flow\\_setSource](#) (void)
- void [unit\\_Flow\\_setDestination](#) (void)
- void [unit\\_Flow\\_getSource](#) (void)
- void [unit\\_Flow\\_getDestination](#) (void)
- void [unit\\_Flow\\_operator](#) (void)
- void [run\\_unit\\_test\\_Flow](#) (void)

### 5.41.1 Function Documentation

#### 5.41.1.1 [run\\_unit\\_test\\_Flow\(\)](#)

```
void run_unit_test_Flow (  
    void )
```

Definition at line 64 of file [unit\\_flow.cpp](#).

#### 5.41.1.2 [unit\\_Flow\\_constructor\(\)](#)

```
void unit_Flow_constructor (  
    void )
```

Definition at line 3 of file [unit\\_flow.cpp](#).

#### 5.41.1.3 unit\_Flow\_destructor()

```
void unit_Flow_destructor (  
    void )
```

Definition at line 10 of file [unit\\_flow.cpp](#).

#### 5.41.1.4 unit\_Flow\_getDestination()

```
void unit_Flow_getDestination (  
    void )
```

Definition at line 49 of file [unit\\_flow.cpp](#).

#### 5.41.1.5 unit\_Flow\_getSource()

```
void unit_Flow_getSource (  
    void )
```

Definition at line 42 of file [unit\\_flow.cpp](#).

#### 5.41.1.6 unit\_Flow\_operator()

```
void unit_Flow_operator (  
    void )
```

Definition at line 56 of file [unit\\_flow.cpp](#).

#### 5.41.1.7 unit\_Flow\_setDestination()

```
void unit_Flow_setDestination (  
    void )
```

Definition at line 33 of file [unit\\_flow.cpp](#).

#### 5.41.1.8 unit\_Flow\_setSource()

```
void unit_Flow_setSource (  
    void )
```

Definition at line 24 of file [unit\\_flow.cpp](#).

## 5.42 unit\_flow.h

[Go to the documentation of this file.](#)

```

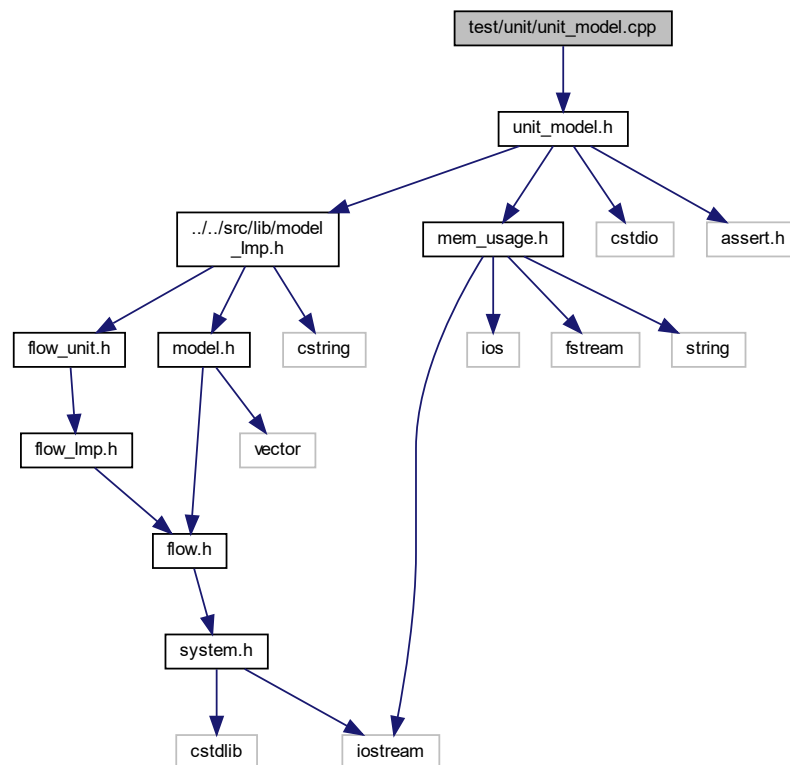
00001 #ifndef UNIT_FLOW_H
00002 #define UNIT_FLOW_H
00003
00004 #include "../src/lib/system_imp.h"
00005 #include "../src/lib/flow_unit.h"
00006 #include "mem_usage.h"
00007
00008 #include <cstdio>
00009 #include <assert.h>
00010
00011 void unit_Flow_constructor(void);
00012 void unit_Flow_destructor(void);
00013 void unit_Flow_setSource(void);
00014 void unit_Flow_setDestination(void);
00015 void unit_Flow_getSource(void);
00016 void unit_Flow_getDestination(void);
00017 void unit_Flow_operator(void);
00018 void run_unit_test_Flow(void);
00019
00020 #endif

```

## 5.43 test/unit/unit\_model.cpp File Reference

```
#include "unit_model.h"
```

Include dependency graph for unit\_model.cpp:



## Functions

- void [unit\\_Model\\_constructor](#) (void)
- void [unit\\_Model\\_destructor](#) (void)
- void [unit\\_Model\\_run](#) (void)
- void [unit\\_Model\\_add\\_System](#) (void)
- void [unit\\_Model\\_add\\_Flow](#) (void)
- void [run\\_unit\\_test\\_Model](#) ()

### 5.43.1 Function Documentation

#### 5.43.1.1 [run\\_unit\\_test\\_Model\(\)](#)

```
void run_unit_test_Model (  
    void )
```

Definition at line 52 of file [unit\\_model.cpp](#).

#### 5.43.1.2 [unit\\_Model\\_add\\_Flow\(\)](#)

```
void unit_Model_add_Flow (  
    void )
```

Definition at line 44 of file [unit\\_model.cpp](#).

#### 5.43.1.3 [unit\\_Model\\_add\\_System\(\)](#)

```
void unit_Model_add_System (  
    void )
```

Definition at line 37 of file [unit\\_model.cpp](#).

#### 5.43.1.4 [unit\\_Model\\_constructor\(\)](#)

```
void unit_Model_constructor (  
    void )
```

Definition at line 3 of file [unit\\_model.cpp](#).

### 5.43.1.5 unit\_Model\_destructor()

```
void unit_Model_destructor (
    void )
```

Definition at line 10 of file [unit\\_model.cpp](#).

### 5.43.1.6 unit\_Model\_run()

```
void unit_Model_run (
    void )
```

Definition at line 24 of file [unit\\_model.cpp](#).

## 5.44 unit\_model.cpp

[Go to the documentation of this file.](#)

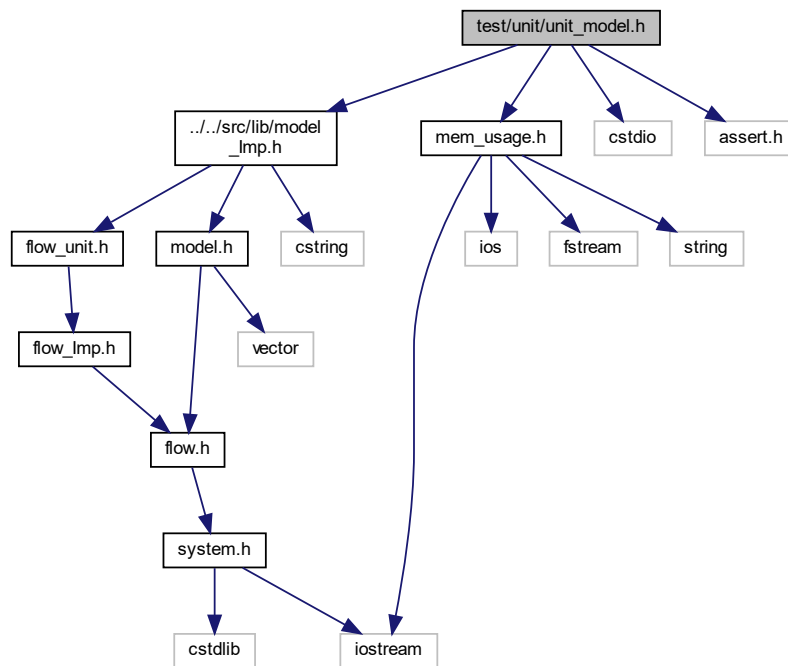
```
00001 #include "unit_model.h"
00002
00003 void unit_Model_constructor(void){
00004     Model* m = Model::createModel("test model");
00005     System* s;
00006     s = m->createSystem("system",10.0 );
00007     assert(m->getSystem("system") == s);
00008 }
00009
00010 void unit_Model_destructor(void){
00011     double memoryBefore, memoryAfter, rss;
00012
00013     mem_usage(memoryBefore, rss);
00014
00015     Model* m = new Model_Imp();
00016     delete m;
00017
00018     mem_usage(memoryAfter, rss);
00019
00020     assert(memoryBefore == memoryAfter);
00021     cout << "Verification: destructor of Model. OK!" << endl;
00022 }
00023
00024 void unit_Model_run(void){
00025     Model* Modelexponential = Model_Imp::createModel("Model pops");
00026     System* pop1;
00027     System* pop2;
00028     Flow* f;
00029     pop1 = Modelexponential->createSystem("pop1", 100.0);
00030     pop2 = Modelexponential->createSystem("pop2", 0.0);
00031     f = Modelexponential->createFlow<Exponential>(pop1,pop2);
00032     Modelexponential->run(0,100);
00033     assert((pop1->getValue() - 36.6032) < 0.0001);
00034     assert((pop2->getValue() - 63.3968) < 0.0001);
00035 }
00036
00037 void unit_Model_add_System(void){
00038     Model* m = Model::createModel("test add");
00039     System* s;
00040     s = m->createSystem("testSystem",0);
00041     assert(m->getSystem("testSystem") == s);
00042 }
00043
00044 void unit_Model_add_Flow(void){
00045     Model* m = Model::createModel("test add");
00046     System* s;
00047     System* s2;
00048     Flow* f = m->createFlow<Logistic>(s,s2);
00049     assert(m->getFlows().empty() == 0);
00050 }
00051
00052 void run_unit_test_Model(){
00053     unit_Model_constructor();
00054     unit_Model_destructor();
00055     unit_Model_run();
00056     unit_Model_add_System();
00057     unit_Model_add_Flow();
00058 }
```



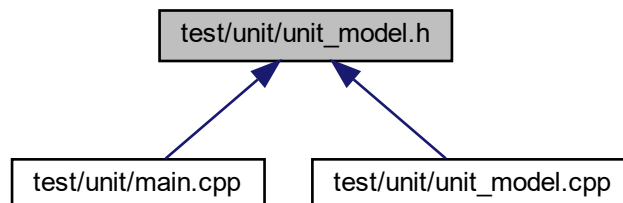
## 5.45 test/unit/unit\_model.h File Reference

```
#include "../src/lib/model_imp.h"
#include "mem_usage.h"
#include <cstdio>
#include <assert.h>
```

Include dependency graph for unit\_model.h:



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



## Functions

- void `unit_Model_constructor` (void)

- void [unit\\_Model\\_destructor](#) (void)
- void [unit\\_Model\\_run](#) (void)
- void [unit\\_Model\\_add\\_System](#) (void)
- void [unit\\_Model\\_add\\_Flow](#) (void)
- void [run\\_unit\\_test\\_Model](#) (void)

## 5.45.1 Function Documentation

### 5.45.1.1 [run\\_unit\\_test\\_Model\(\)](#)

```
void run_unit_test_Model (  
    void )
```

Definition at line [52](#) of file [unit\\_model.cpp](#).

### 5.45.1.2 [unit\\_Model\\_add\\_Flow\(\)](#)

```
void unit_Model_add_Flow (  
    void )
```

Definition at line [44](#) of file [unit\\_model.cpp](#).

### 5.45.1.3 [unit\\_Model\\_add\\_System\(\)](#)

```
void unit_Model_add_System (  
    void )
```

Definition at line [37](#) of file [unit\\_model.cpp](#).

### 5.45.1.4 [unit\\_Model\\_constructor\(\)](#)

```
void unit_Model_constructor (  
    void )
```

Definition at line [3](#) of file [unit\\_model.cpp](#).

#### 5.45.1.5 unit\_Model\_destructor()

```
void unit_Model_destructor (
    void )
```

Definition at line 10 of file [unit\\_model.cpp](#).

#### 5.45.1.6 unit\_Model\_run()

```
void unit_Model_run (
    void )
```

Definition at line 24 of file [unit\\_model.cpp](#).

## 5.46 unit\_model.h

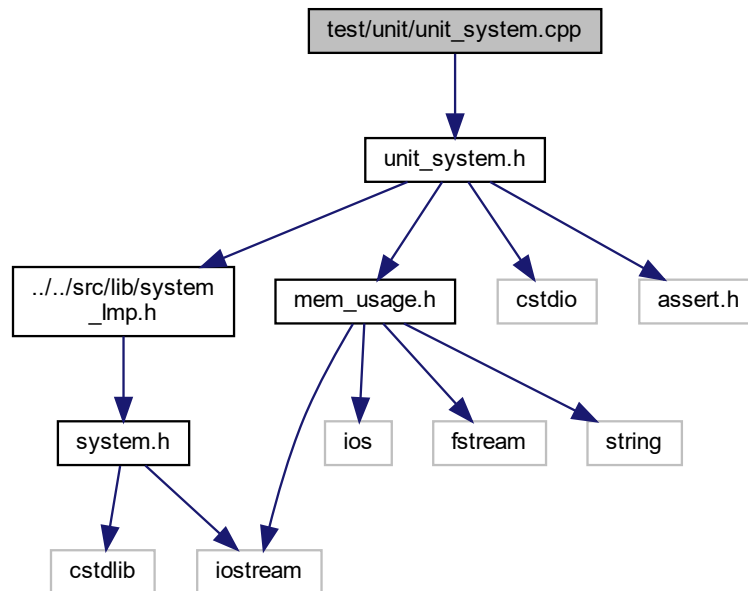
[Go to the documentation of this file.](#)

```
00001 #ifndef UNIT_MODEL_H
00002 #define UNIT_MODEL_H
00003
00004 #include "../src/lib/model_imp.h"
00005 #include "mem_usage.h"
00006
00007 #include <stdio>
00008 #include <assert.h>
00009
00010 void unit_Model_constructor(void);
00011 void unit_Model_destructor(void);
00012 void unit_Model_run(void);
00013 void unit_Model_add_System(void);
00014 void unit_Model_add_Flow(void);
00015 void run_unit_test_Model(void);
00016 #endif
```

## 5.47 test/unit/unit\_system.cpp File Reference

```
#include "unit_system.h"
```

Include dependency graph for unit\_system.cpp:



### Functions

- void [unit\\_System\\_constructor](#) (void)
- void [unit\\_System\\_destructor](#) (void)
- void [unit\\_System\\_setName](#) (void)
- void [unit\\_System\\_setValue](#) (void)
- void [unit\\_System\\_getName](#) (void)
- void [unit\\_System\\_getValue](#) (void)
- void [unit\\_System\\_operator](#) (void)
- void [run\\_unit\\_test\\_System](#) (void)

### 5.47.1 Function Documentation

#### 5.47.1.1 run\_unit\_test\_System()

```
void run_unit_test_System (
    void )
```

Definition at line 54 of file [unit\\_system.cpp](#).

#### 5.47.1.2 unit\_System\_constructor()

```
void unit_System_constructor (  
    void )
```

Definition at line 3 of file [unit\\_system.cpp](#).

#### 5.47.1.3 unit\_System\_destructor()

```
void unit_System_destructor (  
    void )
```

Definition at line 12 of file [unit\\_system.cpp](#).

#### 5.47.1.4 unit\_System\_getName()

```
void unit_System_getName (  
    void )
```

Definition at line 38 of file [unit\\_system.cpp](#).

#### 5.47.1.5 unit\_System\_getValue()

```
void unit_System_getValue (  
    void )
```

Definition at line 43 of file [unit\\_system.cpp](#).

#### 5.47.1.6 unit\_System\_operator()

```
void unit_System_operator (  
    void )
```

Definition at line 48 of file [unit\\_system.cpp](#).

#### 5.47.1.7 unit\_System\_setName()

```
void unit_System_setName (  
    void )
```

Definition at line 26 of file [unit\\_system.cpp](#).

### 5.47.1.8 unit\_System\_setValue()

```
void unit_System_setValue (
    void )
```

Definition at line 32 of file [unit\\_system.cpp](#).

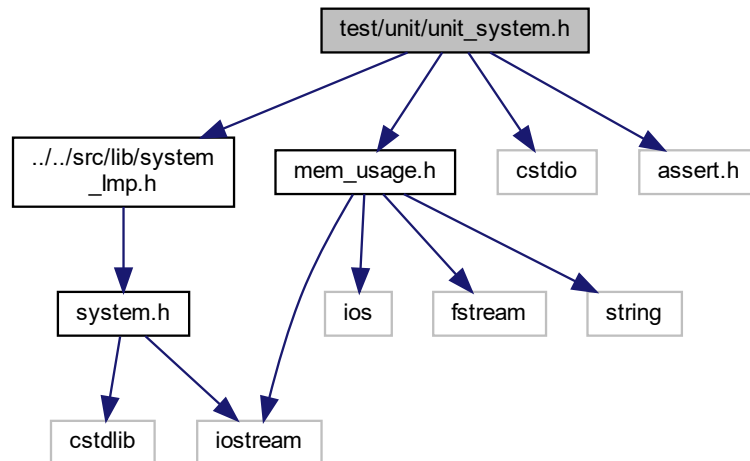
## 5.48 unit\_system.cpp

[Go to the documentation of this file.](#)

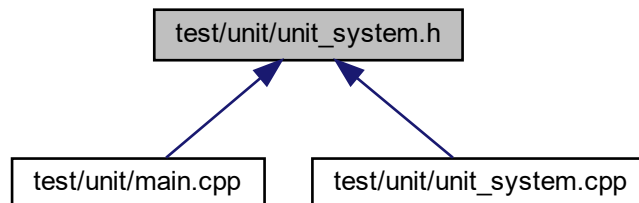
```
00001 #include "unit_system.h"
00002
00003 void unit_System_constructor(void){
00004     System* s1 = new System_Imp();
00005     s1->setValue(0);
00006     assert(s1->getValue() == 0);
00007     System* s2 = new System_Imp();
00008     s2->setValue(10);
00009     assert(s2->getValue() == 10 );
00010 }
00011
00012 void unit_System_destructor(void){
00013     double memoryBefore, memoryAfter, rss;
00014
00015     mem_usage(memoryBefore, rss);
00016
00017     System* s = new System_Imp();
00018     delete s;
00019
00020     mem_usage(memoryAfter, rss);
00021
00022     assert(memoryBefore == memoryAfter);
00023     cout << "Verification: destructor of System. OK!" << endl;
00024 }
00025
00026 void unit_System_setName(void){
00027     System* s1 = new System_Imp();
00028     s1->setName("test");
00029     assert(s1->getName() == "test");
00030 }
00031
00032 void unit_System_setValue(void){
00033     System* s1 = new System_Imp();
00034     s1->setValue(10);
00035     assert(s1->getValue() == 10);
00036 }
00037
00038 void unit_System_getName(void){
00039     System* s1 = new System_Imp("test", 10);
00040     assert(s1->getName() == "test" );
00041 }
00042
00043 void unit_System_getValue(void){
00044     System* s1 = new System_Imp("test", 10);
00045     assert(s1->getValue() == 10 );
00046 }
00047
00048 void unit_System_operator(void){
00049     System* s1 = new System_Imp("test", 10);
00050     System* s2 = s1;
00051     assert(s1->getValue() == s2->getValue());
00052 }
00053
00054 void run_unit_test_System(void){
00055     unit_System_constructor();
00056     unit_System_destructor();
00057     unit_System_setName();
00058     unit_System_setValue();
00059     unit_System_getName();
00060     unit_System_getValue();
00061     unit_System_operator();
00062 }
```

## 5.49 test/unit/unit\_system.h File Reference

```
#include "../src/lib/system_imp.h"
#include "mem_usage.h"
#include <cstdio>
#include <assert.h>
Include dependency graph for unit_system.h:
```



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



### Functions

- void [unit\\_System\\_constructor](#) (void)
- void [unit\\_System\\_destructor](#) (void)
- void [unit\\_System\\_setName](#) (void)
- void [unit\\_System\\_setValue](#) (void)
- void [unit\\_System\\_getName](#) (void)
- void [unit\\_System\\_getValue](#) (void)
- void [unit\\_System\\_operator](#) (void)
- void [run\\_unit\\_test\\_System](#) (void)

## 5.49.1 Function Documentation

### 5.49.1.1 run\_unit\_test\_System()

```
void run_unit_test_System (  
    void )
```

Definition at line 54 of file [unit\\_system.cpp](#).

### 5.49.1.2 unit\_System\_constructor()

```
void unit_System_constructor (  
    void )
```

Definition at line 3 of file [unit\\_system.cpp](#).

### 5.49.1.3 unit\_System\_destructor()

```
void unit_System_destructor (  
    void )
```

Definition at line 12 of file [unit\\_system.cpp](#).

### 5.49.1.4 unit\_System\_getName()

```
void unit_System_getName (  
    void )
```

Definition at line 38 of file [unit\\_system.cpp](#).

### 5.49.1.5 unit\_System\_getValue()

```
void unit_System_getValue (  
    void )
```

Definition at line 43 of file [unit\\_system.cpp](#).



#### 5.49.1.6 unit\_System\_operator()

```
void unit_System_operator (
    void )
```

Definition at line 48 of file [unit\\_system.cpp](#).

#### 5.49.1.7 unit\_System\_setName()

```
void unit_System_setName (
    void )
```

Definition at line 26 of file [unit\\_system.cpp](#).

#### 5.49.1.8 unit\_System\_setValue()

```
void unit_System_setValue (
    void )
```

Definition at line 32 of file [unit\\_system.cpp](#).

## 5.50 unit\_system.h

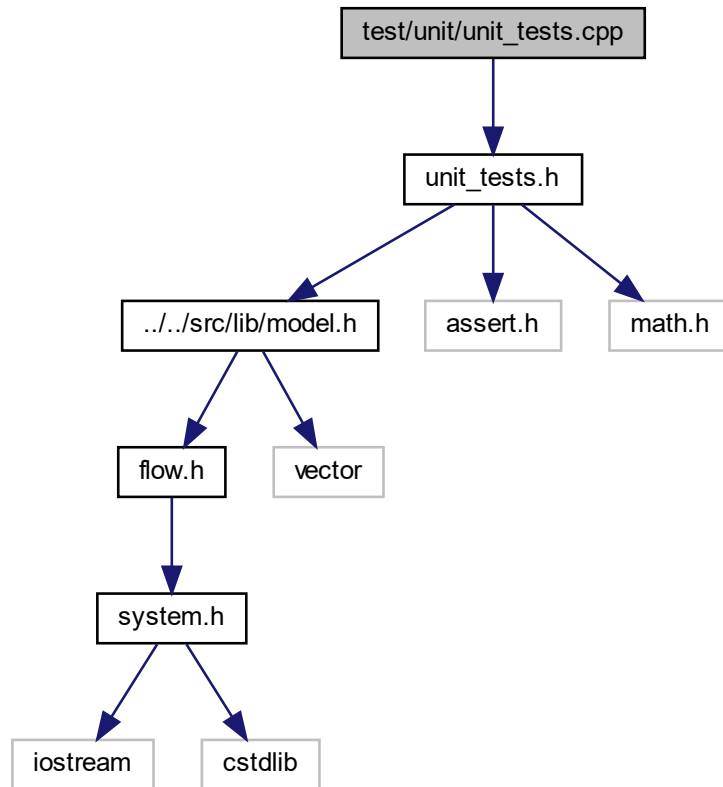
[Go to the documentation of this file.](#)

```
00001 #ifndef UNIT_SYSTEM_H
00002 #define UNIT_SYSTEM_H
00003
00004 #include "../src/lib/system_imp.h"
00005 #include "mem_usage.h"
00006
00007 #include <stdio>
00008 #include <assert.h>
00009
00010
00011 void unit_System_constructor(void);
00012 void unit_System_destructor(void);
00013 void unit_System_setName(void);
00014 void unit_System_setValue(void);
00015 void unit_System_getName(void);
00016 void unit_System_getValue(void);
00017 void unit_System_operator(void);
00018 void run_unit_test_System(void);
00019
00020 #endif
```

## 5.51 test/unit/unit\_tests.cpp File Reference

```
#include "unit_tests.h"
```

Include dependency graph for unit\_tests.cpp:



## 5.52 unit\_tests.cpp

[Go to the documentation of this file.](#)

```
00001 #include "unit_tests.h"
00002
```

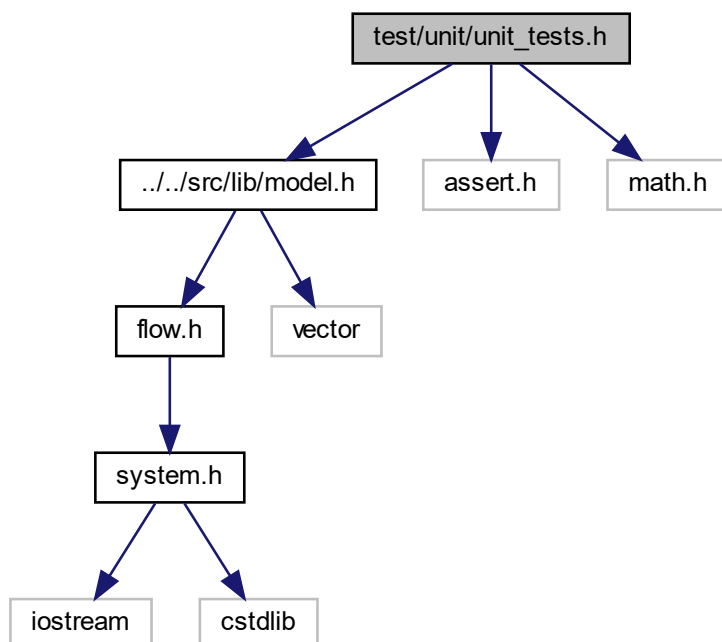
## 5.53 test/unit/unit\_tests.h File Reference

```
#include "../../src/lib/model.h"
```

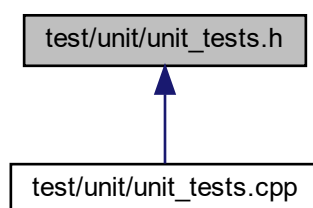
```
#include <assert.h>
```

```
#include <math.h>
```

Include dependency graph for unit\_tests.h:



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



## 5.54 unit\_tests.h

[Go to the documentation of this file.](#)

```

00001 #ifndef UNIT_TESTS_H
00002 #define UNIT_TESTS_H
00003 #include "../../src/lib/model.h"
00004
00005 #include <assert.h>
00006 #include <math.h>
00007
00008 #endif

```



# Index

- ~Flow
  - Flow, [10](#)
- ~Flow\_Imp
  - Flow\_Imp, [14](#)
- ~Model
  - Model, [22](#)
- ~Model\_Imp
  - Model\_Imp, [26](#)
- ~System
  - System, [29](#)
- ~System\_Imp
  - System\_Imp, [33](#)
- complexFuncionalTest
  - functional\_tests.cpp, [57](#)
  - functional\_tests.h, [60](#)
- createFlow
  - Model, [22](#)
- createModel
  - Model, [22](#)
  - Model\_Imp, [26](#)
- createSystem
  - Model, [23](#)
  - Model\_Imp, [26](#)
- destination
  - Flow\_Imp, [17](#)
- Exponential, [7](#)
  - Exponential, [8](#)
  - run, [9](#)
- exponentialFuncionalTest
  - functional\_tests.cpp, [57](#)
  - functional\_tests.h, [60](#)
- Flow, [9](#)
  - ~Flow, [10](#)
  - getDestination, [10](#)
  - getSource, [11](#)
  - operator=, [11](#)
  - run, [11](#)
  - setDestination, [12](#)
  - setSources, [12](#)
- Flow\_Imp, [12](#)
  - ~Flow\_Imp, [14](#)
  - destination, [17](#)
  - Flow\_Imp, [14](#)
  - getDestination, [15](#)
  - getSource, [15](#)
  - operator=, [15](#)
  - run, [16](#)
  - setDestination, [16](#)
  - setSources, [16](#)
  - source, [17](#)
- flows
  - Model\_Imp, [28](#)
- FlowUnit, [18](#)
- functional\_tests.cpp
  - complexFuncionalTest, [57](#)
  - exponentialFuncionalTest, [57](#)
  - logisticalFuncionalTest, [58](#)
- functional\_tests.h
  - complexFuncionalTest, [60](#)
  - exponentialFuncionalTest, [60](#)
  - logisticalFuncionalTest, [60](#)
- getDestination
  - Flow, [10](#)
  - Flow\_Imp, [15](#)
- getFlows
  - Model, [23](#)
  - Model\_Imp, [27](#)
- getName
  - System, [30](#)
  - System\_Imp, [34](#)
- getSource
  - Flow, [11](#)
  - Flow\_Imp, [15](#)
- getSystem
  - Model, [23](#)
  - Model\_Imp, [27](#)
- getValue
  - System, [30](#)
  - System\_Imp, [34](#)
- Logistic, [19](#)
  - Logistic, [20](#)
  - run, [20](#)
- logisticalFuncionalTest
  - functional\_tests.cpp, [58](#)
  - functional\_tests.h, [60](#)
- main
  - main.cpp, [53–55](#)
- main.cpp
  - main, [53–55](#)
- mem\_usage
  - mem\_usage.cpp, [62](#)
  - mem\_usage.h, [63](#)
- mem\_usage.cpp

- mem\_usage, 62
- mem\_usage.h
  - mem\_usage, 63
- Model, 21
  - ~Model, 22
  - createFlow, 22
  - createModel, 22
  - createSystem, 23
  - getFlows, 23
  - getSystem, 23
  - operator=, 23
  - run, 24
- Model\_Imp, 24
  - ~Model\_Imp, 26
  - createModel, 26
  - createSystem, 26
  - flows, 28
  - getFlows, 27
  - getSystem, 27
  - Model\_Imp, 26
  - models, 28
  - operator=, 27
  - run, 27
  - systems, 28
- models
  - Model\_Imp, 28
- name
  - System\_Imp, 35
- operator=
  - Flow, 11
  - Flow\_Imp, 15
  - Model, 23
  - Model\_Imp, 27
  - System, 30
  - System\_Imp, 34
- run
  - Exponential, 9
  - Flow, 11
  - Flow\_Imp, 16
  - Logistic, 20
  - Model, 24
  - Model\_Imp, 27
- run\_unit\_test\_Flow
  - unit\_flow.cpp, 64
  - unit\_flow.h, 68
- run\_unit\_test\_Model
  - unit\_model.cpp, 71
  - unit\_model.h, 74
- run\_unit\_test\_System
  - unit\_system.cpp, 76
  - unit\_system.h, 80
- setDestination
  - Flow, 12
  - Flow\_Imp, 16
- setName
  - System, 31
  - System\_Imp, 35
- setSources
  - Flow, 12
  - Flow\_Imp, 16
- setValue
  - System, 31
  - System\_Imp, 35
- source
  - Flow\_Imp, 17
- src/lib/flow.h, 37, 38
- src/lib/flow\_Imp.cpp, 39
- src/lib/flow\_Imp.h, 40, 41
- src/lib/flow\_unit.h, 42, 43
- src/lib/model.h, 43, 44
- src/lib/model\_Imp.cpp, 45
- src/lib/model\_Imp.h, 47, 48
- src/lib/mySim.cpp, 49
- src/lib/mySlim.h, 49
- src/lib/system.h, 49, 50
- src/lib/system\_Imp.cpp, 50, 51
- src/lib/system\_Imp.h, 51, 52
- src/main.cpp, 52, 53
- System, 29
  - ~System, 29
  - getName, 30
  - getValue, 30
  - operator=, 30
  - setName, 31
  - setValue, 31
- System\_Imp, 31
  - ~System\_Imp, 33
  - getName, 34
  - getValue, 34
  - name, 35
  - operator=, 34
  - setName, 35
  - setValue, 35
  - System\_Imp, 33
  - value, 35
- systems
  - Model\_Imp, 28
- test/functional/functional\_tests.cpp, 56, 58
- test/functional/functional\_tests.h, 59, 61
- test/functional/main.cpp, 54, 55
- test/unit/main.cpp, 55, 56
- test/unit/mem\_usage.cpp, 61, 62
- test/unit/mem\_usage.h, 62, 63
- test/unit/unit\_flow.cpp, 64, 66
- test/unit/unit\_flow.h, 67, 70
- test/unit/unit\_model.cpp, 70, 72
- test/unit/unit\_model.h, 73, 75
- test/unit/unit\_system.cpp, 76, 78
- test/unit/unit\_system.h, 79, 81
- test/unit/unit\_tests.cpp, 82
- test/unit/unit\_tests.h, 82, 83
- unit\_flow.cpp

- run\_unit\_test\_Flow, 64
- unit\_Flow\_constructor, 65
- unit\_Flow\_destructor, 65
- unit\_Flow\_getDestination, 65
- unit\_Flow\_getSource, 65
- unit\_Flow\_operator, 65
- unit\_Flow\_setDestination, 65
- unit\_Flow\_setSource, 66
- unit\_flow.h
  - run\_unit\_test\_Flow, 68
  - unit\_Flow\_constructor, 68
  - unit\_Flow\_destructor, 68
  - unit\_Flow\_getDestination, 69
  - unit\_Flow\_getSource, 69
  - unit\_Flow\_operator, 69
  - unit\_Flow\_setDestination, 69
  - unit\_Flow\_setSource, 69
- unit\_Flow\_constructor
  - unit\_flow.cpp, 65
  - unit\_flow.h, 68
- unit\_Flow\_destructor
  - unit\_flow.cpp, 65
  - unit\_flow.h, 68
- unit\_Flow\_getDestination
  - unit\_flow.cpp, 65
  - unit\_flow.h, 69
- unit\_Flow\_getSource
  - unit\_flow.cpp, 65
  - unit\_flow.h, 69
- unit\_Flow\_operator
  - unit\_flow.cpp, 65
  - unit\_flow.h, 69
- unit\_Flow\_setDestination
  - unit\_flow.cpp, 65
  - unit\_flow.h, 69
- unit\_Flow\_setSource
  - unit\_flow.cpp, 66
  - unit\_flow.h, 69
- unit\_model.cpp
  - run\_unit\_test\_Model, 71
  - unit\_Model\_add\_Flow, 71
  - unit\_Model\_add\_System, 71
  - unit\_Model\_constructor, 71
  - unit\_Model\_destructor, 71
  - unit\_Model\_run, 72
- unit\_model.h
  - run\_unit\_test\_Model, 74
  - unit\_Model\_add\_Flow, 74
  - unit\_Model\_add\_System, 74
  - unit\_Model\_constructor, 74
  - unit\_Model\_destructor, 74
  - unit\_Model\_run, 75
- unit\_Model\_add\_Flow
  - unit\_model.cpp, 71
  - unit\_model.h, 74
- unit\_Model\_add\_System
  - unit\_model.cpp, 71
  - unit\_model.h, 74
- unit\_Model\_constructor
  - unit\_model.cpp, 71
  - unit\_model.h, 74
- unit\_Model\_destructor
  - unit\_model.cpp, 71
  - unit\_model.h, 74
- unit\_Model\_run
  - unit\_model.cpp, 72
  - unit\_model.h, 75
- unit\_system.cpp
  - run\_unit\_test\_System, 76
  - unit\_System\_constructor, 76
  - unit\_System\_destructor, 77
  - unit\_System\_getName, 77
  - unit\_System\_getValue, 77
  - unit\_System\_operator, 77
  - unit\_System\_setName, 77
  - unit\_System\_setValue, 77
- unit\_system.h
  - run\_unit\_test\_System, 80
  - unit\_System\_constructor, 80
  - unit\_System\_destructor, 80
  - unit\_System\_getName, 80
  - unit\_System\_getValue, 80
  - unit\_System\_operator, 80
  - unit\_System\_setName, 81
  - unit\_System\_setValue, 81
- unit\_System\_constructor
  - unit\_system.cpp, 76
  - unit\_system.h, 80
- unit\_System\_destructor
  - unit\_system.cpp, 77
  - unit\_system.h, 80
- unit\_System\_getName
  - unit\_system.cpp, 77
  - unit\_system.h, 80
- unit\_System\_getValue
  - unit\_system.cpp, 77
  - unit\_system.h, 80
- unit\_System\_operator
  - unit\_system.cpp, 77
  - unit\_system.h, 80
- unit\_System\_setName
  - unit\_system.cpp, 77
  - unit\_system.h, 81
- unit\_System\_setValue
  - unit\_system.cpp, 77
  - unit\_system.h, 81
- value
  - System\_Imp, 35