

# MIN17212 - Projet Simulation - Documentation

1.0

Généré par Doxygen 1.8.13

## Table des matières

<b>1</b>	<b>Index des classes</b>	<b>2</b>
1.1	Liste des classes . . . . .	2
<b>2</b>	<b>Index des fichiers</b>	<b>2</b>
2.1	Liste des fichiers . . . . .	2
<b>3</b>	<b>Documentation des classes</b>	<b>2</b>
3.1	Référence de la structure <code>s_echeancier</code> . . . . .	2
3.1.1	Description détaillée . . . . .	3
3.1.2	Documentation des données membres . . . . .	3
3.2	Référence de la structure <code>s_evenement</code> . . . . .	4
3.2.1	Description détaillée . . . . .	4
3.2.2	Documentation des données membres . . . . .	4
3.3	Référence de la structure <code>s_simulation</code> . . . . .	5
3.3.1	Description détaillée . . . . .	6
3.3.2	Documentation des données membres . . . . .	6
<b>4</b>	<b>Documentation des fichiers</b>	<b>7</b>
4.1	Référence du fichier <code>src/constantes.h</code> . . . . .	7
4.1.1	Documentation des macros . . . . .	7
4.2	Référence du fichier <code>src/echeancier.c</code> . . . . .	9
4.2.1	Documentation des fonctions . . . . .	9
4.3	Référence du fichier <code>src/echeancier.h</code> . . . . .	11
4.3.1	Documentation des définitions de type . . . . .	12
4.3.2	Documentation du type de l'énumération . . . . .	12
4.3.3	Documentation des fonctions . . . . .	13
4.4	Référence du fichier <code>src/lora.c</code> . . . . .	15
4.4.1	Documentation des fonctions . . . . .	15
4.5	Référence du fichier <code>src/lora.h</code> . . . . .	17
4.5.1	Documentation des définitions de type . . . . .	18
4.5.2	Documentation des fonctions . . . . .	18
4.6	Référence du fichier <code>src/macros.h</code> . . . . .	20
4.6.1	Documentation des macros . . . . .	20
4.7	Référence du fichier <code>src/main.c</code> . . . . .	20
4.7.1	Documentation des fonctions . . . . .	21
4.8	Référence du fichier <code>src/utils.c</code> . . . . .	22
4.8.1	Documentation des fonctions . . . . .	22
4.9	Référence du fichier <code>src/utils.h</code> . . . . .	23
4.9.1	Documentation des fonctions . . . . .	23

<a href="#">Index</a>	25
-----------------------	----

## 1 Index des classes

### 1.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

<a href="#">s_echeancier</a>	
Représente un ensemble d'évènement en fonction de leur date	2
<a href="#">s_evenement</a>	
Représente un évènement dans l'échéancier	4
<a href="#">s_simulation</a>	
La structure qui contient les variables de la simulation	5

## 2 Index des fichiers

### 2.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

<a href="#">src/constantes.h</a>	7
<a href="#">src/echeancier.c</a>	9
<a href="#">src/echeancier.h</a>	11
<a href="#">src/lora.c</a>	15
<a href="#">src/lora.h</a>	17
<a href="#">src/macros.h</a>	20
<a href="#">src/main.c</a>	20
<a href="#">src/utils.c</a>	22
<a href="#">src/utils.h</a>	23

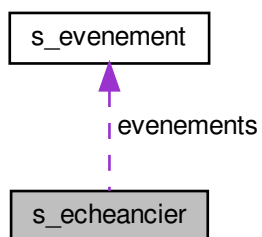
## 3 Documentation des classes

### 3.1 Référence de la structure s\_echeancier

Représente un ensemble d'évènement en fonction de leur date.

```
#include <echeancier.h>
```

Graphe de collaboration de s\_echeancier :



#### Attributs publics

- [Evenement evenements](#) [[MAX\\_EVENTS](#)]  
*Le stockage des événements.*
- [int n](#)  
*Le nombre d'événements actuellement dans l'échéancier.*

#### 3.1.1 Description détaillée

Représente un ensemble d'évènement en fonction de leur date.

#### 3.1.2 Documentation des données membres

##### 3.1.2.1 evenements

[Evenement](#) s\_echeancier::evenements [[MAX\\_EVENTS](#)]

Le stockage des événements.

##### 3.1.2.2 n

`int s_echeancier::n`

Le nombre d'événements actuellement dans l'échéancier.

La documentation de cette structure a été générée à partir du fichier suivant :

- [src/echeancier.h](#)

## 3.2 Référence de la structure s\_evenement

Représente un évènement dans l'échéancier.

```
#include <echeancier.h>
```

### Attributs publics

- [TypeEvenement](#) type  
*Le type d'évènement.*
- int [k](#)  
*L'indentifiant du capteur.*
- int [etat](#)  
*L'état du capteur.*
- double [date](#)  
*La date de l'évènement.*

### 3.2.1 Description détaillée

Représente un évènement dans l'échéancier.

### 3.2.2 Documentation des données membres

#### 3.2.2.1 date

```
double s_evenement::date
```

La date de l'évènement.

#### 3.2.2.2 etat

```
int s_evenement::etat
```

L'état du capteur.

#### 3.2.2.3 k

```
int s_evenement::k
```

L'indentifiant du capteur.

## 3.2.2.4 type

```
TypeEvenement s_evenement::type
```

Le type d'événement.

La documentation de cette structure a été générée à partir du fichier suivant :

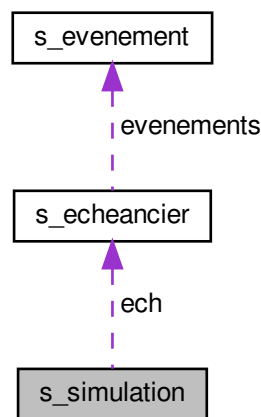
— src/[echeancier.h](#)

## 3.3 Référence de la structure s\_simulation

La structure qui contient les variables de la simulation.

```
#include <lora.h>
```

Graphe de collaboration de s\_simulation :



## Attributs publics

- [Echeancier ech](#)  
*L'échéancier des événements.*
- int [nbTotalEmissions](#) [MAX\_K]  
*Le nombre total d'émissions réussies dans chaque capteur.*
- double [T](#)  
*Le temps de simulation.*
- int [K](#)  
*Le nombre de capteurs dans le réseau.*
- long [nbMinEmissions](#)  
*Nombre de capteurs qui ont atteint le minimum d'émissions réussies.*
- long [nbCollisions](#) [MAX\_ESSAIS]  
*Nombre de collisions dans chaque état.*
- long [nbEmissions](#) [MAX\_ESSAIS]  
*Nombre d'émissions totales dans chaque état.*
- double [tempsEmission](#) [MAX\_ESSAIS]  
*Temps d'émission observée dans chaque état.*

### 3.3.1 Description détaillée

La structure qui contient les variables de la simulation.

### 3.3.2 Documentation des données membres

#### 3.3.2.1 ech

```
Echeancier s_simulation::ech
```

L'échéancier des évènements.

#### 3.3.2.2 K

```
int s_simulation::K
```

Le nombre de capteurs dans le réseau.

#### 3.3.2.3 nbCollisions

```
long s_simulation::nbCollisions[MAX_ESSAIS]
```

Nombre de collisions dans chaque état.

#### 3.3.2.4 nbEmissions

```
long s_simulation::nbEmissions[MAX_ESSAIS]
```

Nombre d'émissions totales dans chaque état.

#### 3.3.2.5 nbMinEmissions

```
long s_simulation::nbMinEmissions
```

Nombre de capteurs qui ont atteint le minimum d'émissions réussies.

### 3.3.2.6 nbTotalEmissions

```
int s_simulation::nbTotalEmissions[MAX_K]
```

Le nombre total d'émissions réussies dans chaque capteur.

### 3.3.2.7 T

```
double s_simulation::T
```

Le temps de simulation.

### 3.3.2.8 tempsEmission

```
double s_simulation::tempsEmission[MAX_ESSAIS]
```

Temps d'émission observée dans chaque état.

La documentation de cette structure a été générée à partir du fichier suivant :

— src/[lora.h](#)

## 4 Documentation des fichiers

### 4.1 Référence du fichier src/constantes.h

#### Macros

- #define [LAMBDA\\_E](#) 10  
*Lambda de la durée d'émission d'un paquet.*
- #define [LAMBDA\\_I](#) 0.1  
*Lambda du temps d'attente après une émission réussie.*
- #define [LAMBDA\\_W](#) 0.25  
*Lambda du temps d'attente en cas d'échec.*
- #define [MAX\\_ESSAIS](#) 7  
*Le nombre maximum d'essais d'émission avant que le paquet soit perdu.*
- #define [MAX\\_EVENEMENTS](#) 1000  
*Le nombre maximum d'évènements dans l'échéancier.*
- #define [MAX\\_K](#) 100  
*Le nombre maximum de capteurs dans le réseau.*
- #define [MIN\\_EMISSIONS](#) 1e3  
*Le nombre minimum de messages correctement émis par chaque capteur avant que la simulation soit terminée.*

#### 4.1.1 Documentation des macros



**4.1.1.1 LAMBDA\_E**

```
#define LAMBDA_E 10
```

Lambda de la durée d'émission d'un paquet.

**4.1.1.2 LAMBDA\_I**

```
#define LAMBDA_I 0.1
```

Lambda du temps d'attente après une émission réussie.

**4.1.1.3 LAMBDA\_W**

```
#define LAMBDA_W 0.25
```

Lambda du temps d'attente en cas d'échec.

**4.1.1.4 MAX\_ESSAIS**

```
#define MAX_ESSAIS 7
```

Le nombre maximum d'essais d'émission avant que le paquet soit perdu.

**4.1.1.5 MAX\_EVENEMENTS**

```
#define MAX_EVENEMENTS 1000
```

Le nombre maximum d'évènements dans l'échéancier.

**4.1.1.6 MAX\_K**

```
#define MAX_K 100
```

Le nombre maximum de capteurs dans le réseau.

**4.1.1.7 MIN\_EMISSIONS**

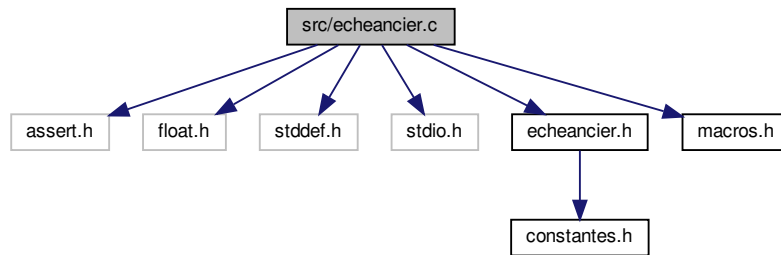
```
#define MIN_EMISSIONS 1e3
```

Le nombre minimum de messages correctement émis par chaque capteur avant que la simulation soit terminée.

## 4.2 Référence du fichier src/echeancier.c

```
#include <assert.h>
#include <float.h>
#include <stddef.h>
#include <stdio.h>
#include "echeancier.h"
#include "macros.h"
```

Graphe des dépendances par inclusion de echeancier.c :



### Fonctions

- void `echeancier_ajouter` (`Echeancier *e`, `TypeEvenement type`, `int k`, `int etat`, `double date`)  
*Ajoute un événement à l'échéancier.*
- `Evenement * echeancier_detector_collision` (`Echeancier *e`, `double date`)  
*Détecte si un événement est en collision avec un autre dans l'échéancier.*
- `Evenement echeancier_suivant` (`Echeancier *e`)  
*Récupère le prochain événement de l'échéancier.*
- void `echeancier_init` (`Echeancier *e`)  
*Initialise l'échéancier.*
- int `echeancier_vide` (`const Echeancier *e`)  
*Vérifie si l'échéancier est vide.*

### 4.2.1 Documentation des fonctions

#### 4.2.1.1 echeancier\_ajouter()

```
void echeancier_ajouter (
    Echeancier * e,
    TypeEvenement type,
    int k,
    int etat,
    double date )
```

Ajoute un événement à l'échéancier.

#### Paramètres

<i>e</i>	L'échéancier.
<i>type</i>	Le type de l'événement.
<i>k</i>	L'identifiant du capteur.
<i>etat</i>	L'état du capteur.
<i>date</i>	La date de l'événement.

#### 4.2.1.2 echeancier\_detector\_collision()

```
Evenement* echeancier_detector_collision (
    Echeancier * e,
    double date )
```

Détecte si un évènement est en collision avec un autre dans l'échéancier.

##### Paramètres

<i>e</i>	L'échéancier.
<i>date</i>	La date de l'évènement à tester.

##### Renvoie

L'évènement en collision avec e1, ou NULL si aucun évènement n'est en collision.

#### 4.2.1.3 echeancier\_init()

```
void echeancier_init (
    Echeancier * e )
```

Initialise l'échéancier.

##### Paramètres

<i>e</i>	L'échéancier.
----------	---------------

#### 4.2.1.4 echeancier\_suivant()

```
Evenement echeancier_suivant (
    Echeancier * e )
```

Récupère le prochain événement de l'échéancier.

Cet évènement est supprimé de l'échéancier. Le comportement est indéfini si l'échéancier est vide.

##### Paramètres

<i>e</i>	L'échéancier.
----------	---------------

##### Renvoie

Le prochain événement.

Voir également

[echeancier\\_vide](#)

#### 4.2.1.5 echeancier\_vide()

```
int echeancier_vide (
    const Echeancier * e )
```

Vérifie si l'échéancier est vide.

Paramètres

<i>e</i>	L'échéancier.
----------	---------------

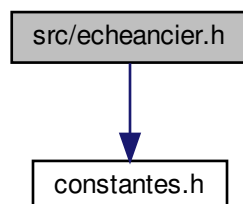
Renvoie

1 si l'échéancier est vide, 0 sinon.

### 4.3 Référence du fichier src/echeancier.h

```
#include "constantes.h"
```

Graphe des dépendances par inclusion de echeancier.h :



Classes

- struct [s\\_evenement](#)  
*Représente un évènement dans l'échéancier.*
- struct [s\\_echeancier](#)  
*Représente un ensemble d'évènement en fonction de leur date.*

Définitions de type

- typedef enum [e\\_type\\_evenement](#) TypeEvenement
- typedef struct [s\\_evenement](#) Evenement  
*Représente un évènement dans l'échéancier.*
- typedef struct [s\\_echeancier](#) Echeancier  
*Représente un ensemble d'évènement en fonction de leur date.*

## Énumérations

- enum `e_type_evenement` { DE, FE, TC }

## Fonctions

- void `echancierajouter` (`Echancier` \*e, `TypeEvenement` type, int k, int etat, double date)  
*Ajoute un événement à l'échéancier.*
- `Evenement` \* `echancierdetecter_collision` (`Echancier` \*e, double date)  
*Détecte si un événement est en collision avec un autre dans l'échéancier.*
- `Evenement` `echancier_suivant` (`Echancier` \*e)  
*Récupère le prochain événement de l'échéancier.*
- void `echancier_init` (`Echancier` \*e)  
*Initialise l'échéancier.*
- int `echancier_vide` (const `Echancier` \*e)  
*Vérifie si l'échéancier est vide.*

### 4.3.1 Documentation des définitions de type

#### 4.3.1.1 Echancier

```
typedef struct s_echancier Echancier
```

Représente un ensemble d'évènement en fonction de leur date.

#### 4.3.1.2 Evenement

```
typedef struct s_evenement Evenement
```

Représente un événement dans l'échéancier.

#### 4.3.1.3 TypeEvenement

```
typedef enum e_type_evenement TypeEvenement
```

### 4.3.2 Documentation du type de l'énumération

#### 4.3.2.1 e\_type\_evenement

```
enum e_type_evenement
```

## Valeurs énumérées

DE	Début émission.
FE	Fin émission.
TC	Traitement collision.

## 4.3.3 Documentation des fonctions

## 4.3.3.1 echeancier\_ajouter()

```
void echeancier_ajouter (
    Echeancier * e,
    TypeEvenement type,
    int k,
    int etat,
    double date )
```

Ajoute un événement à l'échéancier.

## Paramètres

<i>e</i>	L'échéancier.
<i>type</i>	Le type de l'événement.
<i>k</i>	L'indentifiant du capteur.
<i>etat</i>	L'état du capteur.
<i>date</i>	La date de l'événement.

## 4.3.3.2 echeancier\_detecter\_collision()

```
Evenement* echeancier_detecter_collision (
    Echeancier * e,
    double date )
```

Détecte si un évènement est en collision avec un autre dans l'échéancier.

## Paramètres

<i>e</i>	L'échéancier.
<i>date</i>	La date de l'événement à tester.

## Renvoie

L'évènement en collision avec e1, ou NULL si aucun évènement n'est en collision.

#### 4.3.3.3 echeancier\_init()

```
void echeancier_init (
    Echeancier * e )
```

Initialise l'échéancier.

##### Paramètres

<i>e</i>	L'échéancier.
----------	---------------

#### 4.3.3.4 echeancier\_suivant()

```
Evenement echeancier_suivant (
    Echeancier * e )
```

Récupère le prochain événement de l'échéancier.

Cet événement est supprimé de l'échéancier. Le comportement est indéfini si l'échéancier est vide.

##### Paramètres

<i>e</i>	L'échéancier.
----------	---------------

##### Renvoie

Le prochain événement.

##### Voir également

[echeancier\\_vide](#)

#### 4.3.3.5 echeancier\_vide()

```
int echeancier_vide (
    const Echeancier * e )
```

Vérifie si l'échéancier est vide.

##### Paramètres

<i>e</i>	L'échéancier.
----------	---------------

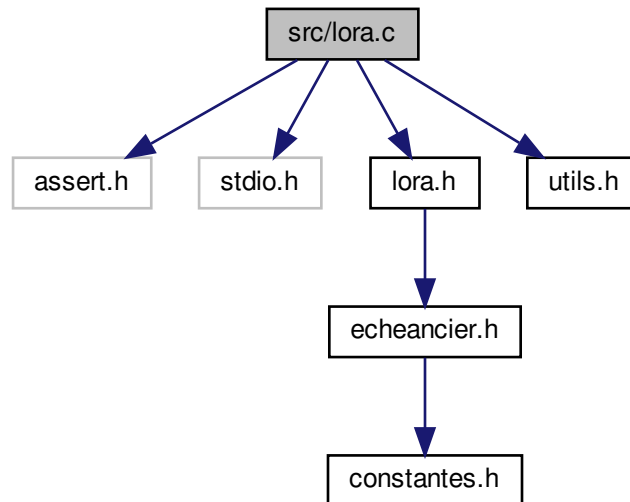
##### Renvoie

1 si l'échéancier est vide, 0 sinon.

#### 4.4 Référence du fichier src/lora.c

```
#include <assert.h>
#include <stdio.h>
#include "lora.h"
#include "utils.h"
```

Graphe des dépendances par inclusion de lora.c :



#### Fonctions

- void `Traitement_Event` (`Simulation *sim`, const `Evenement *e`)  
*Traite un événement.*
- void `Traitement_Collision` (`Simulation *sim`, const `Evenement *e1`, `Evenement *e2`)  
*Traite une collision.*
- void `Simulateur` (`Simulation *sim`, int showAll)  
*Exécute une simulation.*
- void `simulation_init` (`Simulation *sim`, int K)  
*Initialise la simulation.*
- void `simulation_print` (const `Simulation *sim`)  
*Affiche l'état de la simulation.*

#### 4.4.1 Documentation des fonctions

##### 4.4.1.1 Simulateur()

```
void Simulateur (
    Simulation * sim,
    int showAll )
```

Exécute une simulation.



**Paramètres**

<i>sim</i>	La simulation.
<i>showAll</i>	0 pour n'afficher que la fin de la simulation, sinon affiche toutes les étapes.

**4.4.1.2 simulation\_init()**

```
void simulation_init (
    Simulation * sim,
    int K )
```

Initialise la simulation.

**Paramètres**

<i>sim</i>	La structure de la simulation à initialiser.
<i>K</i>	Le nombre de capteurs dans le réseau.

**4.4.1.3 simulation\_print()**

```
void simulation_print (
    const Simulation * sim )
```

Affiche l'état de la simulation.

**Paramètres**

<i>sim</i>	La simulation à afficher.
------------	---------------------------

**4.4.1.4 Traitement\_Collision()**

```
void Traitement_Collision (
    Simulation * sim,
    const Evenement * e1,
    Evenement * e2 )
```

Traite une collision.

**Paramètres**

<i>sim</i>	La simulation.
<i>e1</i>	L'événement à traiter.
<i>e2</i>	L'événement à venir en collision.

## 4.4.1.5 Traitement\_Event()

```
void Traitement_Event (
    Simulation * sim,
    const Evenement * e )
```

Traite un événement.

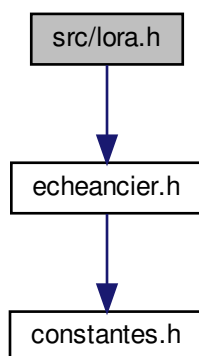
## Paramètres

<i>sim</i>	La simulation.
<i>e</i>	L'événement à traiter.

## 4.5 Référence du fichier src/lora.h

```
#include "echeancier.h"
```

Graphe des dépendances par inclusion de lora.h :



## Classes

- struct `s_simulation`  
*La structure qui contient les variables de la simulation.*

## Définitions de type

- typedef struct `s_simulation` `Simulation`  
*La structure qui contient les variables de la simulation.*

## Fonctions

- void `Traitement_Event` (`Simulation` \*sim, const `Evenement` \*e)  
*Traite un événement.*
- void `Traitement_Collision` (`Simulation` \*sim, const `Evenement` \*e1, `Evenement` \*e2)  
*Traite une collision.*
- void `Simulateur` (`Simulation` \*sim, int showAll)  
*Exécute une simulation.*
- void `simulation_init` (`Simulation` \*sim, int K)  
*Initialise la simulation.*
- void `simulation_print` (const `Simulation` \*sim)  
*Affiche l'état de la simulation.*

### 4.5.1 Documentation des définitions de type

#### 4.5.1.1 Simulation

```
typedef struct s_simulation Simulation
```

La structure qui contient les variables de la simulation.

### 4.5.2 Documentation des fonctions

#### 4.5.2.1 Simulateur()

```
void Simulateur (
    Simulation * sim,
    int showAll )
```

Exécute une simulation.

#### Paramètres

<i>sim</i>	La simulation.
<i>showAll</i>	0 pour n'afficher que la fin de la simulation, sinon affiche toutes les étapes.

#### 4.5.2.2 simulation\_init()

```
void simulation_init (
    Simulation * sim,
    int K )
```

Initialise la simulation.

## Paramètres

<i>sim</i>	La structure de la simulation à initialiser.
<i>K</i>	Le nombre de capteurs dans le réseau.

## 4.5.2.3 simulation\_print()

```
void simulation_print (
    const Simulation * sim )
```

Affiche l'état de la simulation.

## Paramètres

<i>sim</i>	La simulation à afficher.
------------	---------------------------

## 4.5.2.4 Traitement\_Collision()

```
void Traitement_Collision (
    Simulation * sim,
    const Evenement * e1,
    Evenement * e2 )
```

Traite une collision.

## Paramètres

<i>sim</i>	La simulation.
<i>e1</i>	L'événement à traiter.
<i>e2</i>	L'évènement à venir en collision.

## 4.5.2.5 Traitement\_Event()

```
void Traitement_Event (
    Simulation * sim,
    const Evenement * e )
```

Traite un événement.

## Paramètres

<i>sim</i>	La simulation.
<i>e</i>	L'événement à traiter.

## 4.6 Référence du fichier src/macros.h

### Macros

- `#define SWAP(A, B)`  
*Echange le contenu de deux variables.*

### 4.6.1 Documentation des macros

#### 4.6.1.1 SWAP

```
#define SWAP(  
    A,  
    B )
```

#### Valeur :

```
do { \
    typeof(A) C = (A); \
    (A) = (B); \
    (B) = C; \
} while (0)
```

Echange le contenu de deux variables.

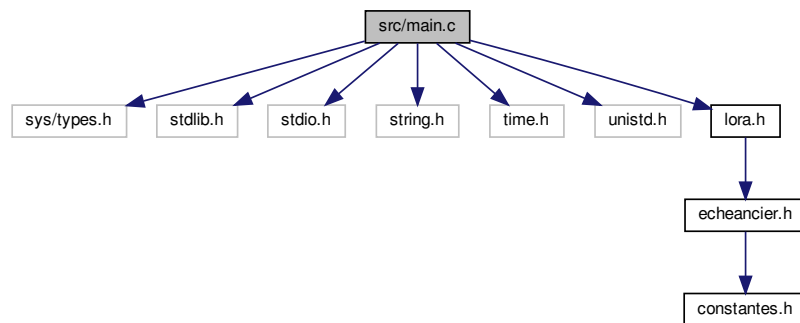
#### Paramètres

<i>A</i>	La première variable.
<i>B</i>	La seconde variable.

## 4.7 Référence du fichier src/main.c

```
#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "lora.h"
```

Graphe des dépendances par inclusion de main.c :



### Fonctions

- static int `print_error` (const char \*section, const char \*message)
- static int `print_usage` (char \*bin)
- int `main` (int ac, char \*\*av)

#### 4.7.1 Documentation des fonctions

##### 4.7.1.1 main()

```
int main (  
    int ac,  
    char ** av )
```

##### 4.7.1.2 print\_error()

```
static int print_error (  
    const char * section,  
    const char * message ) [static]
```

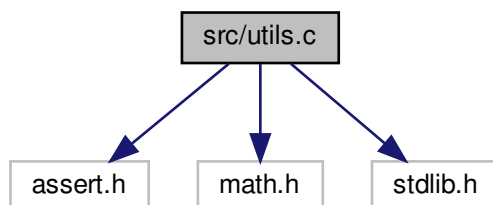
##### 4.7.1.3 print\_usage()

```
static int print_usage (  
    char * bin ) [static]
```

## 4.8 Référence du fichier src/Utils.c

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

Graphes des dépendances par inclusion de Utils.c :



### Fonctions

- static double `rand_double` ()  
*Génère un nombre aléatoire entre 0 et 1.*
- double `Expo_Duree` (double `lambda`)  
*Génère un nombre aléatoire suivant une loi exponentielle.*

### 4.8.1 Documentation des fonctions

#### 4.8.1.1 `Expo_Duree()`

```
double Expo_Duree (
    double lambda )
```

Génère un nombre aléatoire suivant une loi exponentielle.

Le générateur de nombres pseudo-aléatoires doit être initialisé avec `srandom()`.

#### Paramètres

<code>lambda</code>	La constante de la loi exponentielle.
---------------------	---------------------------------------

#### Renvoie

Le valeur tirée aléatoirement.

## 4.8.1.2 rand\_double()

```
static double rand_double ( ) [static]
```

Génère un nombre aléatoire entre 0 et 1.

## Renvoie

Le nombre tirée aléatoirement.

## 4.9 Référence du fichier src/Utils.h

## Fonctions

- double [Expo\\_Duree](#) (double lambda)  
*Génère un nombre aléatoire suivant une loi exponentielle.*

## 4.9.1 Documentation des fonctions

## 4.9.1.1 Expo\_Duree()

```
double Expo_Duree (
    double lambda )
```

Génère un nombre aléatoire suivant une loi exponentielle.

Le générateur de nombres pseudo-aléatoires doit être initialisé avec srandom().

## Paramètres

<i>lambda</i>	La constante de la loi exponentielle.
---------------	---------------------------------------

## Renvoie

Le valeur tirée aléatoirement.





## Index

constantes.h  
    LAMBDA\_E, 7  
    LAMBDA\_I, 8  
    LAMBDA\_W, 8  
    MAX\_ESSAIS, 8  
    MAX\_EVENEMENTS, 8  
    MAX\_K, 8  
    MIN\_EMISSIONS, 8

date  
    s\_evenement, 4

e\_type\_evenement  
    echeancier.h, 12

ech  
    s\_simulation, 6

Echeancier  
    echeancier.h, 12

echeancier.c  
    echeancier\_ajouter, 9  
    echeancier\_detector\_collision, 10  
    echeancier\_init, 10  
    echeancier\_suivant, 10  
    echeancier\_vide, 11

echeancier.h  
    e\_type\_evenement, 12  
    Echeancier, 12  
    echeancier\_ajouter, 13  
    echeancier\_detector\_collision, 13  
    echeancier\_init, 13  
    echeancier\_suivant, 14  
    echeancier\_vide, 14  
    Evenement, 12  
    TypeEvenement, 12

echeancier\_ajouter  
    echeancier.c, 9  
    echeancier.h, 13

echeancier\_detector\_collision  
    echeancier.c, 10  
    echeancier.h, 13

echeancier\_init  
    echeancier.c, 10  
    echeancier.h, 13

echeancier\_suivant  
    echeancier.c, 10  
    echeancier.h, 14

echeancier\_vide  
    echeancier.c, 11  
    echeancier.h, 14

etat  
    s\_evenement, 4

Evenement  
    echeancier.h, 12

evenements  
    s\_echeancier, 3

Expo\_Duree  
    utils.c, 22  
    utils.h, 23

K  
    s\_simulation, 6

k  
    s\_evenement, 4

LAMBDA\_E  
    constantes.h, 7

LAMBDA\_I  
    constantes.h, 8

LAMBDA\_W  
    constantes.h, 8

lora.c  
    Simulateur, 15  
    simulation\_init, 16  
    simulation\_print, 16  
    Traitement\_Collision, 16  
    Traitement\_Event, 16

lora.h  
    Simulateur, 18  
    Simulation, 18  
    simulation\_init, 18  
    simulation\_print, 19  
    Traitement\_Collision, 19  
    Traitement\_Event, 19

MAX\_ESSAIS  
    constantes.h, 8

MAX\_EVENEMENTS  
    constantes.h, 8

MAX\_K  
    constantes.h, 8

MIN\_EMISSIONS  
    constantes.h, 8

macros.h  
    SWAP, 20

main  
    main.c, 21

main.c  
    main, 21  
    print\_error, 21  
    print\_usage, 21

n  
    s\_echeancier, 3

nbCollisions  
    s\_simulation, 6

nbEmissions  
    s\_simulation, 6

nbMinEmissions  
    s\_simulation, 6

nbTotalEmissions  
    s\_simulation, 6

print\_error

- main.c, [21](#)
- print\_usage
  - main.c, [21](#)
- rand\_double
  - utils.c, [22](#)
- s\_echeancier, [2](#)
  - evenements, [3](#)
  - n, [3](#)
- s\_evenement, [4](#)
  - date, [4](#)
  - etat, [4](#)
  - k, [4](#)
  - type, [4](#)
- s\_simulation, [5](#)
  - ech, [6](#)
  - K, [6](#)
  - nbCollisions, [6](#)
  - nbEmissions, [6](#)
  - nbMinEmissions, [6](#)
  - nbTotalEmissions, [6](#)
  - T, [7](#)
  - tempsEmission, [7](#)
- SWAP
  - macros.h, [20](#)
- Simulateur
  - lora.c, [15](#)
  - lora.h, [18](#)
- Simulation
  - lora.h, [18](#)
- simulation\_init
  - lora.c, [16](#)
  - lora.h, [18](#)
- simulation\_print
  - lora.c, [16](#)
  - lora.h, [19](#)
- src/constantes.h, [7](#)
- src/echeancier.c, [9](#)
- src/echeancier.h, [11](#)
- src/lora.c, [15](#)
- src/lora.h, [17](#)
- src/macros.h, [20](#)
- src/main.c, [20](#)
- src/utils.c, [22](#)
- src/utils.h, [23](#)
- T
  - s\_simulation, [7](#)
- tempsEmission
  - s\_simulation, [7](#)
- Traitement\_Collision
  - lora.c, [16](#)
  - lora.h, [19](#)
- Traitement\_Event
  - lora.c, [16](#)
  - lora.h, [19](#)
- type
  - s\_evenement, [4](#)
- TypeEvenement
  - echeancier.h, [12](#)
- utils.c
  - Expo\_Duree, [22](#)
  - rand\_double, [22](#)
- utils.h
  - Expo\_Duree, [23](#)