

libjournal

1.1.57

Généré par Doxygen 1.8.9.1

Vendredi 25 Septembre 2015 08 :03 :37

Table des matières

1	Index des fichiers	1
1.1	Liste des fichiers	1
2	Documentation des fichiers	1
2.1	Référence du fichier lib/journal.h	1
2.1.1	Documentation des macros	2
2.1.2	Documentation du type de l'énumération	2
2.1.3	Documentation des fonctions	3
2.1.4	Documentation des variables	4
	Index	5

1 Index des fichiers

1.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

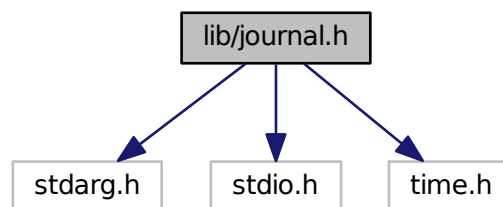
lib/journal.h **1**

2 Documentation des fichiers

2.1 Référence du fichier lib/journal.h

```
#include <stdarg.h>
#include <stdio.h>
#include <time.h>
```

Graphe des dépendances par inclusion de journal.h :



Macros

— #define **JOURNAL**(niveau, args...) **ecrire_entree_journal**(niveau, __FILE__, __LINE__, args)

Énumérations

```
— enum niveau_t {
    NONE = 0, ERROR = 1, WARNING = 2, INFO = 3,
    DEBUG = 4 }
```

Fonctions

```
— int ouvrir_journal (niveau_t niveau, char *nom_fichier, FILE *fichier, int ajouter)
— void fermer_journal (void)
— void ecrire_entree_journal (niveau_t niveau, char *fichier, int ligne, char *format,...)
```

Variables

```
— static char * niveau_vers_chaine []
```

2.1.1 Documentation des macros

2.1.1.1 #define JOURNAL(*niveau*, *args...*) `ecrire_entree_journal`(niveau, __FILE__, __LINE__, args)

MACRO permettant de faciliter l'ajout d'une entrée au journal.

Cette MACRO fournit une interface simplifiée à `ecrire_entree_journal()` en passant les définitions `__FILE__` et `__LINE__` correspondant toutes deux au **nom du fichier** appelant la MACRO et le **numéro de ligne** d'appel à la MACRO, respectivement.

Exemple :

```
if(ouvrir_journal(ERROR, "mon_journal.log", NULL, 0) != 0)
{
    return 1;
}
else{
    char* message = "Bonjour le monde !";
    JOURNAL(DEBUG, "Le message de DEBUG est : %s", message);
}
```

Paramètres

<i>niveau</i>	Le niveau de criticité de l'entrée.
<i>args</i>	La chaîne de caractères spécifiant le message à journaliser (accompagnée des variables si nécessaire).

Voir également

`ecrire_entree_journal()`

2.1.2 Documentation du type de l'énumération

2.1.2.1 enum `niveau_t`

Énumération des différents niveaux de criticité de journalisation.

Chaque énumération peut être convertie en chaîne de caractère grâce à la structure de données `niveau_vers_chaine[]`.

Voir également

`niveau_vers_chaine[]`

Valeurs énumérées

`NONE`

ERROR**WARNING****INFO****DEBUG**

```
15     {
16     NONE = 0,
17     ERROR = 1,
18     WARNING = 2,
19     INFO = 3,
20     DEBUG = 4,
21 } niveau_t;
```

2.1.3 Documentation des fonctions

2.1.3.1 void `ecrire_entree_journal` (`niveau_t` `niveau`, `char *` *fichier*, `int` *ligne*, `char *` *format*, ...)

Écrit une entrée dans le fichier de journalisation.

Ne peut être appelé SI ET SEULEMENT SI `ouvrir_journal` a été appelé et n'a pas retourné d'erreur.

Exemple :

```
char* message = "Bonjour le monde !";
ecrire_entree_journal(DEBUG, __FILE__, __LINE__, "Le message de DEBUG est : %s",
    message);
```

Paramètres

<i>niveau</i>	Le niveau de criticité de l'entrée à journaliser
<i>fichier</i>	Une chaîne décrivant le fichier réalisant l'entrée. <code>__FILE__</code> est sans nul doute ce que vous préféreriez saisir.
<i>ligne</i>	Un nombre décrivant le numéro de la ligne réalisant l'entrée. <code>__LINE__</code> est sans nul doute ce que vous préféreriez saisir.
<i>format</i>	La chaîne de caractères spécifiant le message à journaliser formatée comme une 'printf format string'.

Voir également

[JOURNAL](#)

2.1.3.2 void `fermer_journal` (void)

Termine l'utilisation possible de la journalisation.

Clos le fichier de journalisation et empêche tout ajout d'une nouvelle entrée au journal.

Exemple :

```
if(ouvrir_journal(ERROR, "mon_journal.log", NULL, 0) != 0)
{
    return 1;
}
else{
    // Faire quelque chose
    // ...
    fermer_journal();
}
```

Voir également

[ouvrir_journal\(\)](#)

2.1.3.3 `int ouvrir_journal (niveau_t niveau, char * nom_fichier, FILE * fichier, int ajouter)`

Initialise la journalisation.

Exemple :

```
int resultat;
resultat = ouvrir_journal(ERROR, "mon_journal.log", NULL, 0);
```

ou bien

```
int resultat;
FILE* fichier_journal_perso;
fichier_journal_perso = fopen("mon_journal.log", "w");
resultat = ouvrir_journal(ERROR, NULL, fichier_journal_perso, 0);
```

Ouvrira, par écrasement, le fichier nommé `mon_journal.log` pour y écrire tout événement de criticité inférieur ou égale à `ERROR`.

Paramètres

<i>niveau</i>	Tout futur appel à <code>ecrire_entree_journal()</code> ne journalisera qu'à un niveau inférieur ou égal à ce niveau de criticité.
<i>nom_fichier</i>	La chaîne de caractères définissant le chemin du fichier de journalisation (si <code>NULL</code> , alors fichier doit être valorisé).
<i>fichier</i>	Pointeur sur <code>FILE</code> d'un fichier déjà déclaré (si <code>NULL</code> , alors <i>nom_fichier</i> doit être valorisé).
<i>ajouter</i>	Mode ajout au fichier si différent de 0, mode écrasé si égal à 0.

Renvoie

0 si tout va bien, -1 sinon

Voir également

[ecrire_entree_journal](#)

2.1.4 Documentation des variables

2.1.4.1 `char* niveau_vers_chaine[]` `[static]`

Valeur initiale :

```
= {
    NULL,
    "ERROR",
    "WARNING",
    "INFO",
    "DEBUG",
}
```

Tableau de chaînes assurant la conversion : niveau de criticité => texte

Index

DEBUG

journal.h, [3](#)

ERROR

journal.h, [2](#)

ecrire_entree_journal

journal.h, [3](#)

fermer_journal

journal.h, [3](#)

INFO

journal.h, [3](#)

JOURNAL

journal.h, [2](#)

journal.h

DEBUG, [3](#)

ERROR, [2](#)

ecrire_entree_journal, [3](#)

fermer_journal, [3](#)

INFO, [3](#)

JOURNAL, [2](#)

NONE, [2](#)

niveau_t, [2](#)

niveau_vers_chaine, [4](#)

ouvrir_journal, [3](#)

WARNING, [3](#)

lib/journal.h, [1](#)

NONE

journal.h, [2](#)

niveau_t

journal.h, [2](#)

niveau_vers_chaine

journal.h, [4](#)

ouvrir_journal

journal.h, [3](#)

WARNING

journal.h, [3](#)