Documentation utilisateur de l'outil de documentation des programmes en langage C

Dernière modification : 18/02/2024

Auteur : Ewen JAIN

Sommaire

- `#define / constantes`
- `Structures`

Introduction à l'outil

Bienvenue dans la documentation de notre outil de documentation pour programme C. Cette ressource vous guidera dans l'utilisisation efficace de cet outil avec des instructions de mise en place et d'utilisation, afin d'optimiser vos processus de documentation.

Pourquoi utiliser notre produit ?

Notre outil de documentation a été concu dans l'objectif de simplifier et d'améliorer significativement votre processus de documentation. Vous obtiendrez une organisation structurée de votre documentation en fonction des commentaires analysés afin de produire un rendu propre et professionel, améliorant la compréhension de votre code pour vous, votre équipe et vos clients.

Comment utiliser notre produit ?

Notre générateur de documentation est simple d'utilisation, placez au minimum 1 fichier C dans le même dossier que le script bash, ainsi qu'un fichier markdown.

Ensuite, executez simplement le script "gendoc.bash" (après l'avoir rendu executable)

Le script génère une archive avec comme nom nom_client-version.tar.gz, ainsi qu'un dossier "doss_archive" actualisé à chaque execution, avec le même contenu a la seule différence qu'il n'est pas archivé.

Vous pouvez récupérer votre archive ou le dossier à l'emplacement où vous avez lancé le script Bash.

Il vous suffira ensuite d'extraire ou de récupérer le contenu du dossier, et vous y trouverez le(s) fichier(s) C originaux, ainsi que leur documentations, au formats HTML et PDF, en plus d'une documentation utilisateur de cet outil (c'est vrai que ce n'est pas nécéssaire ici, mais c'etait demandé de générer une doc .md -> .html -> .pdf et que ce sera donc la même que celle que vous lisez actuellement, mais en html et en pdf...).

Documentation du code

Le code a analyser doit respecter certaines contraintes (notamment de syntaxe) afin de génerer une documentation cohérente.

En-tête de programme

Chacun de vos fichiers doivent impérativement débuter par un commentaire en en-tête, et ce dès la 1ere ligne, afin que l'outil de documentation puisse analyser la suite du programme.

L'en-tête, comme tout commentaire multi-ligne, aura la forme suivante:

```
/**
* Ce bloc est obligatoire !
* Ici devra être rédigé la description
* du contenu du fichier, ce que fait le
* programme en globalité.
*/
```

Documentation des fonctions / procédures

balise	description
\brief	courte phrase pour décrire **brièvement** le rôle de la fonction
\detail	une description plus détaillée du role de la fonction et de son utilisation
\return	explication du rôle de l'élément retourné par la fonction (sauf si c'est une procédure)
\param	une balise différente doit être utilisée pour chaque paremètre d'entrée (et de sortie pour les procédures), elle détaillera donc le rôle de ces paramètre dans la fonction

Documentation des autres éléments

La syntaxe des commentaires ne change que peu en fonction des éléments concernés, de la forme : `/** commentaire */` .

Celle des `fonctions et procédure` est plus complexe que celle des `autres éléments`, afin d'avoir toutes les informations nécessaires a propos des fonctions concernées, grace a l'utilisation des balises, qui vont donc générer un rendu plus précis dans la documentation.

Pour une variable globale

Pour qu'elles puissent correctement être prises en compte, les variables globales **doivent** être précédées d'un commentaire contenant "VARIABLES GLOBALES",

puis une ligne vide doit être laissée après les avoir toutes définies.

```
 /**VARIABLES GLOBALES*/
int nbLignes = 0; /** nombre de lignes du tableau */
int nbColonnes = 0; /** nombre de colonnes du tableau */
```

Pour un #define ou une simple constante

Ces commentaires sont semblables à ceux des variables, mais ne nécessitent pas de commentaire particulier les précédant.

```
 #define TAILLE 9 /** Définition de la taille des éléments
```

Pour une structure

Pour commenter une structure, il faudra plusieurs commentaires : un après chaque élément de la structure, puis un après avoir fini sa définition, pour donner son rôle en globalité.

```
 typedef struct {
    char prenom[25];    /** prenom de l'utilisateur */
    char pseudo[25];    /** pseudo de l'utilisateur */
    int age;          /** age de l'utilisateur */
} utilisateur;   /** Structure d'un utilisateur */
```