

Tomahawk

v.alpha

Généré par Doxygen 1.8.11

Table des matières

1	Page d'accueil	1
2	Index des structures de données	3
2.1	Structures de données	3
3	Index des fichiers	5
3.1	Liste des fichiers	5
4	Documentation des structures de données	7
4.1	Référence de la structure HTTP_Node	7
4.1.1	Description détaillée	7
4.1.2	Documentation des champs	7
4.1.2.1	beg	7
4.1.2.2	childs	7
4.1.2.3	end	7
4.1.2.4	name	8
4.1.2.5	nb_childs	8

5	Documentation des fichiers	9
5.1	Référence du fichier src/api.h	9
5.1.1	Description détaillée	9
5.1.2	Documentation des fonctions	9
5.1.2.1	copierChaine(char *buf, char *res, int beg, int end)	9
5.1.2.2	parser(char *buf, unsigned int len, char *search, void(*callback)())	10
5.2	Référence du fichier src/http_node.h	10
5.2.1	Description détaillée	11
5.2.2	Documentation des fonctions	11
5.2.2.1	addChild_HTTP_Node(HTTP_Node *node, HTTP_Node *child)	11
5.2.2.2	found_HTTP_Node(HTTP_Node *root, char *string)	11
5.2.2.3	foundAll_HTTP_Node_rec(HTTP_Node *root, char *string, HTTP_Node **result, int *nbFound)	12
5.2.2.4	free_HTTP_Node(HTTP_Node *node)	12
5.2.2.5	free_HTTP_Tree(HTTP_Node *root)	12
5.2.2.6	init_HTTP_Node(char *name, HTTP_Node *node)	12
5.2.2.7	print_HTTP_Node(char *request, HTTP_Node *node)	12
5.2.2.8	print_HTTP_Tree(char *request, HTTP_Node *root, int level)	13
5.3	Référence du fichier src/http_parser.h	13
5.3.1	Description détaillée	14
5.3.2	Documentation des fonctions	14
5.3.2.1	isDIGIT(char *request, int *cursor)	14
5.3.2.2	isFieldvchar(char *request, int *cursor)	15
5.3.2.3	isObstext(char *request, int *cursor)	15
5.3.2.4	isTchar(char *request, int *cursor)	15
5.3.2.5	isUnreserved(char *request, int *cursor)	15
5.3.2.6	isVCHAR(char *request, int *cursor)	16
5.3.2.7	parse_absolute_form(char *request, int *cursor, HTTP_Node *node)	16
5.3.2.8	parse_authority_form(char *request, int *cursor, HTTP_Node *node)	16
5.3.2.9	parse_field_content(char *request, int *cursor, HTTP_Node *header_field, HT↔ TP_Node *field_name, HTTP_Node *field_value)	17

5.3.2.10	<code>parse_field_name(char *request, int *cursor, HTTP_Node *header_field, HTTP↔ P_Node *field_name, HTTP_Node *field_value)</code>	17
5.3.2.11	<code>parse_field_value(char *request, int *cursor, HTTP_Node *header_field, HTTP↔ P_Node *field_name, HTTP_Node *field_value)</code>	17
5.3.2.12	<code>parse_header_field(char *request, int *cursor, HTTP_Node *header_field)</code>	18
5.3.2.13	<code>parse_HTTP_message(char *request, int *cursor, HTTP_Node *node)</code>	18
5.3.2.14	<code>parse_IPv4(char *request, int *cursor, HTTP_Node *node)</code>	18
5.3.2.15	<code>parse_message_body(char *request, int *cursor, HTTP_Node *node)</code>	19
5.3.2.16	<code>parse_obs_fold(char *request, int *cursor, HTTP_Node *header_field, HTTP↔ Node *field_name, HTTP_Node *field_value)</code>	19
5.3.2.17	<code>parse_origin_form(char *request, int *cursor, HTTP_Node *node)</code>	19
5.3.2.18	<code>parse_regname(char *request, int *cursor, HTTP_Node *node)</code>	20
5.3.2.19	<code>parse_request_line(char *request, int *cursor, HTTP_Node *node)</code>	20
5.3.2.20	<code>parse_request_target(char *request, int *cursor, HTTP_Node *node)</code>	20
5.3.2.21	<code>parse_start_line(char *request, int *cursor, HTTP_Node *node)</code>	21
5.3.2.22	<code>parse_status_line(char *request, int *cursor, HTTP_Node *node)</code>	21
5.3.2.23	<code>parse_string(char *str, int *cursor, char *cmp_str)</code>	21
5.3.2.24	<code>parse_user_info(char *request, int *cursor, HTTP_Node *node)</code>	22

Chapitre 1

Page d'accueil

Le projet Tomahawk est développé et maintenu comme un serveur HTTP open-source pour les systèmes d'exploitation moderne, ce qui inclut UNIX et Windows. Le but étant de procurer un serveur peu sécurisé, peu fiable, moyennement efficace et conforme aux standards HTTP du siècle dernier.

Le serveur Tomahawk

Ce serveur sera bientôt un incontournable pour toutes les entreprises.

L'équipe de développement

L'équipe de développement est fantastique, et mettra tout en oeuvre pour maintenir le développement de Tomahawk.

Doués, passionnés et impliqués ils sont prêts à recevoir toutes remarques et répondront avec plaisir au mail d'admirateur. Comme le dis Dennis MacAlistair Ritchie à leurs propos : "Ces mecs sont formidables".

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

HTTP_Node	
Représente un noeud de l'arbre utilisé par le parseur	7

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

src/ api.h	API du Parseur HTTP 1.0	9
src/ http_node.h	Gestion de l'arbre n-aire utiliser par le parseur HTTP	10
src/ http_parser.h	Parseur HTTP 1.0	13

Chapitre 4

Documentation des structures de données

4.1 Référence de la structure HTTP_Node

Représente un noeud de l'arbre utilisé par le parseur.

```
#include <http_node.h>
```

Champs de données

- char * `name`
- int `beg`
- int `end`
- HTTP_Node ** `childs`
- int `nb_childs`

4.1.1 Description détaillée

Représente un noeud de l'arbre utilisé par le parseur.

4.1.2 Documentation des champs

4.1.2.1 int beg

Début de l'élément dans la requête HTTP.

4.1.2.2 HTTP_Node** childs

Ensemble de ses enfants.

4.1.2.3 int end

Fin de l'élément dans la requête HTTP.

4.1.2.4 `char* name`

Nom du noeud.

4.1.2.5 `int nb_childs`

Nombre d'enfant.

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

— `src/http_node.h`

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier src/api.h

API du Parseur HTTP 1.0.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Fonctions

- int **parser** (char *buf, unsigned int len, char *search, void(*callback>())
Fonction demande par l'API de test.
- char * **copierChaine** (char *buf, char *res, int beg, int end)
Fonction permettant de recopier une sous-chaine.

5.1.1 Description détaillée

API du Parseur HTTP 1.0.

Auteur

R. Vidal / A. Lorber / T. Jacumin

Version

1.0

5.1.2 Documentation des fonctions

5.1.2.1 char* copierChaine (char * buf, char * res, int beg, int end)

Fonction permettant de recopier une sous-chaine.

Paramètres

<i>buf</i>	Pointeur vers le premier caractère de la chaîne.
<i>res</i>	Pointeur vers le résultat.
<i>beg</i>	Début de la sous-chaîne.
<i>end</i>	Fin de la sous-chaîne.

Renvoie

Un pointeur vers *res*.

5.1.2.2 int parser (char * *buf*, unsigned int *len*, char * *search*, void(*)() *callback*)

Fonction demandée par l'API de test.

Paramètres

<i>Buf</i>	pointeur vers le premier caractère du flux.
<i>Len</i>	longueur du flux à traiter.
<i>Search</i>	champs à chercher.
<i>Callback</i>	fonction de callback.

Renvoie

-1 si le message est valide syntaxiquement, sinon un entier correspondant à l'indice dans la chaîne *buf* ou le parser à détecter une erreur syntaxique.

5.2 Référence du fichier src/http_node.h

Gestion de l'arbre n-aire utilisé par le parseur HTTP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Structures de données

- struct [HTTP_Node](#)
Représente un nœud de l'arbre utilisé par le parseur.

Définitions de type

- typedef struct [HTTP_Node](#) **HTTP_Node**

Fonctions

- void `init_HTTP_Node` (char *name, `HTTP_Node` *node)
Fonction de création d'une nouvelle instance d'un objet `HTTP_Node`.
- void `addChild_HTTP_Node` (`HTTP_Node` *node, `HTTP_Node` *child)
Fonction d'ajout d'un fils à un noeud.
- void `free_HTTP_Node` (`HTTP_Node` *node)
Fonction de suppression d'un noeud.
- void `free_HTTP_Tree` (`HTTP_Node` *root)
Fonction de suppression d'un arbre.
- `HTTP_Node` * `found_HTTP_Node` (`HTTP_Node` *root, char *string)
Fonction de recherche dans un arbre.
- void `foundAll_HTTP_Node` (`HTTP_Node` *root, char *string, int *nbFound, `HTTP_Node` **result)
- void `foundAll_HTTP_Node_rec` (`HTTP_Node` *root, char *string, `HTTP_Node` **result, int *nbFound)
Fonction de recherche dans un arbre.
- void `print_HTTP_Node` (char *request, `HTTP_Node` *node)
Fonction pour afficher le contenu d'un `HTTP_Node`.
- void `print_HTTP_Tree` (char *request, `HTTP_Node` *root, int level)
Fonction pour afficher le contenu d'un arbre de `HTTP_Node`.

5.2.1 Description détaillée

Gestion de l'arbre n-aire utiliser par le parseur HTTP.

Auteur

R. Vidal / A. Lorber / T. Jacumin

Version

1.0

5.2.2 Documentation des fonctions

5.2.2.1 void addChild_HTTP_Node (HTTP_Node * node, HTTP_Node * child)

Fonction d'ajout d'un fils à un noeud.

Paramètres

<i>node</i>	Pointeur vers le noeud parent.
<i>child</i>	Pointeur vers le noeud fils.

5.2.2.2 HTTP_Node* found_HTTP_Node (HTTP_Node * root, char * string)

Fonction de recherche dans un arbre.

Paramètres

<i>root</i>	Pointeur vers le noeud racine de l'arbre à supprimer.
<i>string</i>	La chaîne à rechercher.

Renvoie

Le noeud qui à pour nom la chaine string.

5.2.2.3 void foundAll_HTTP_Node_rec (HTTP_Node * root, char * string, HTTP_Node ** result, int * nbFound)

Fonction de recherche dans un arbre.

Paramètres

<i>root</i>	Pointeur vers le noeud racine de l'arbre à supprimer.
<i>string</i>	La chaîne à rechercher.
<i>result</i>	Tableau des noeuds correspondant à la recherche.
<i>nbFound</i>	Le nombre de noeud trouvé.

5.2.2.4 void free_HTTP_Node (HTTP_Node * node)

Fonction de suppression d'un noeud.

Paramètres

<i>node</i>	Pointeur vers le noeud à supprimer.
-------------	-------------------------------------

5.2.2.5 void free_HTTP_Tree (HTTP_Node * root)

Fonction de suppression d'un arbre.

Paramètres

<i>node</i>	Pointeur vers le noeud racine de l'arbre à supprimer.
-------------	---

5.2.2.6 void init_HTTP_Node (char * name, HTTP_Node * node)

Fonction de création d'une nouvelle instance d'un objet [HTTP_Node](#).

Paramètres

<i>name</i>	Nom du noeud.
<i>node</i>	Pointeur vers le noeud à initialiser.

5.2.2.7 void print_HTTP_Node (char * request, HTTP_Node * node)

Fonction pour afficher le contenu d'un [HTTP_Node](#).

Paramètres

<i>request</i>	Chaîne contenant la requête.
<i>node</i>	Noeud à afficher.

5.2.2.8 void print_HTTP_Tree (char * *request*, HTTP_Node * *root*, int *level*)

Fonction pour afficher le contenu d'un arbre de [HTTP_Node](#).

Paramètres

<i>request</i>	Chaîne contenant la requête.
<i>node</i>	Noeud racine.
<i>level</i>	Pour l'indentation (laisser à 0).

5.3 Référence du fichier src/http_parser.h

Parseur HTTP 1.0.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "http_node.h"
```

Macros

```
— #define SP ' '
— #define DEBUG
— #define DEBUG_PRINT(x) printf(x)
```

Fonctions

- int [parse_HTTP_message](#) (char **request*, int **cursor*, [HTTP_Node](#) **node*)
Fonction pour parser le HTTP-message.
- int [parse_start_line](#) (char **request*, int **cursor*, [HTTP_Node](#) **node*)
Fonction pour parser le start-line.
- int [parse_status_line](#) (char **request*, int **cursor*, [HTTP_Node](#) **node*)
Fonction pour parser le status-line.
- int [parse_header_field](#) (char **request*, int **cursor*, [HTTP_Node](#) **header_field*)
Fonction pour parser le header-field.
- int [parse_field_name](#) (char **request*, int **cursor*, [HTTP_Node](#) **header_field*, [HTTP_Node](#) **field_name*, [HTTP_Node](#) **field_value*)
Fonction pour parser le field-name.
- int [parse_field_value](#) (char **request*, int **cursor*, [HTTP_Node](#) **header_field*, [HTTP_Node](#) **field_name*, [HTTP_Node](#) **field_value*)
Fonction pour parser le field-content (utilisée par header-field).
- int [parse_field_content](#) (char **request*, int **cursor*, [HTTP_Node](#) **header_field*, [HTTP_Node](#) **field_name*, [HTTP_Node](#) **field_value*)
Fonction pour parser le field-content (utilisée par header-field).

- int `parse_obs_fold` (char *request, int *cursor, HTTP_Node *header_field, HTTP_Node *field_name, HTTP_Node *field_value)
Fonction pour parser le obs-fold (utilisée par header-field).
- int `parse_message_body` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le message-body.
- int `parse_request_target` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le request-target.
- int `parse_request_line` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le request-line.
- int `parse_origin_form` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le origin-form.
- int `parse_absolute_form` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le absolute-form.
- int `parse_authority_form` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le authority-form.
- int `parse_IPv4` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le IPv4.
- int `parse_user_info` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le user-info.
- int `parse_regname` (char *request, int *cursor, HTTP_Node *node)
Fonction pour parser le regname.
- int `parse_string` (char *str, int *cursor, char *cmp_str)
Fonction pour vérifier l'égalité d'une chaîne et d'une sous-chaîne.
- int `isDIGIT` (char *request, int *cursor)
Fonction pour vérifier si on a bien un DIGIT.
- int `isTchar` (char *request, int *cursor)
Fonction pour vérifier si on a bien un TCHAR.
- int `isVCHAR` (char *request, int *cursor)
Fonction pour vérifier si on a bien un VCHAR.
- int `isObstext` (char *request, int *cursor)
Fonction pour vérifier si on a bien un ObsText.
- int `isFieldvchar` (char *request, int *cursor)
Fonction pour vérifier si on a bien un Field-VCHAR.
- int `isUnreserved` (char *request, int *cursor)
Fonction pour vérifier si on a bien un Unreserved.

5.3.1 Description détaillée

Parseur HTTP 1.0.

Auteur

R. Vidal / A. Lorber / T. Jacumin

Version

1.0

5.3.2 Documentation des fonctions

5.3.2.1 int isDIGIT (char * request, int * cursor)

Fonction pour vérifier si on a bien un DIGIT.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.

Renvoie

Retourne 1 si le prochain caractere est un DIGIT, 0 sinon.

5.3.2.2 int isFieldvchar (char * *request*, int * *cursor*)

Fonction pour vérifier si on a bien un Field-VCHAR.

Paramètres

<i>str</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.

Renvoie

Retourne 1 si la caractere de la requête est un field-vchar, 0 sinon.

5.3.2.3 int isObstext (char * *request*, int * *cursor*)

Fonction pour vérifier si on a bien un ObsText.

Paramètres

<i>str</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.

Renvoie

Retourne 1 si le caractere de la requete est un ObsText, 0 sinon.

5.3.2.4 int isTchar (char * *request*, int * *cursor*)

Fonction pour vérifier si on a bien un TCHAR.

Paramètres

<i>str</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.

Renvoie

Retourne 1 si le caractere de la requete est un TCHAR, 0 sinon.

5.3.2.5 int isUnreserved (char * *request*, int * *cursor*)

Fonction pour vérifier si on a bien un Unreserved.

Paramètres

<i>str</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.

Renvoie

Retourne 1 si la caractere de la requête est un Unreserved, 0 sinon.

5.3.2.6 int isVCHAR (char * *request*, int * *cursor*)

Fonction pour vérifier si on a bien un VCHAR.

Paramètres

<i>str</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.

Renvoie

Retourne 1 si le caractere de la requete est un VCHAR, 0 sinon.

5.3.2.7 int parse_absolute_form (char * *request*, int * *cursor*, HTTP_Node * *node*)

Fonction pour parser le absolute-form.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.8 int parse_authority_form (char * *request*, int * *cursor*, HTTP_Node * *node*)

Fonction pour parser le authority-form.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.9 `int parse_field_content (char * request, int * cursor, HTTP_Node * header_field, HTTP_Node * field_name, HTTP_Node * field_value)`

Fonction pour parser le field-content (utilisée par header-field).

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud header-field utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-name utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-value utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.10 `int parse_field_name (char * request, int * cursor, HTTP_Node * header_field, HTTP_Node * field_name, HTTP_Node * field_value)`

Fonction pour parser le field-name.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud header-field utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-name utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-value utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.11 `int parse_field_value (char * request, int * cursor, HTTP_Node * header_field, HTTP_Node * field_name, HTTP_Node * field_value)`

Fonction pour parser le field-content (utilisée par header-field).

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud header-field utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-name utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-value utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.12 `int parse_header_field (char * request, int * cursor, HTTP_Node * header_field)`

Fonction pour parser le header-field.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.13 `int parse_HTTP_message (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le HTTP-message.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.14 `int parse_IPv4 (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le IPv4.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.15 `int parse_message_body (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le message-body.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.16 `int parse_obs_fold (char * request, int * cursor, HTTP_Node * header_field, HTTP_Node * field_name, HTTP_Node * field_value)`

Fonction pour parser le obs-fold (utilisée par header-field).

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud header-field utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-name utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.
<i>node</i>	Pointeur vers le noeud field-value utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.17 `int parse_origin_form (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le origin-form.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoi

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.18 `int parse_regname (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le regname.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoi

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.19 `int parse_request_line (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le request-line.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoi

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.20 `int parse_request_target (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le request-target.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.21 `int parse_start_line (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le start-line.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.22 `int parse_status_line (char * request, int * cursor, HTTP_Node * node)`

Fonction pour parser le status-line.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.23 `int parse_string (char * str, int * cursor, char * cmp_str)`

Fonction pour vérifier l'égalité d'une chaîne et d'une sous-chaîne.

Paramètres

<i>str</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>cmp_str</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.

5.3.2.24 int parse_user_info (char * *request*, int * *cursor*, HTTP_Node * *node*)

Fonction pour parser le user-info.

Paramètres

<i>request</i>	chaîne contenant la requête.
<i>cursor</i>	Pointeur vers la position de la lecture de la requête.
<i>node</i>	Pointeur vers le noeud utilisé pour sauvegarder la valeur de l'élément correspondant au noeud.

Renvoie

Retourne 1 si la requête est correcte, 0 sinon.